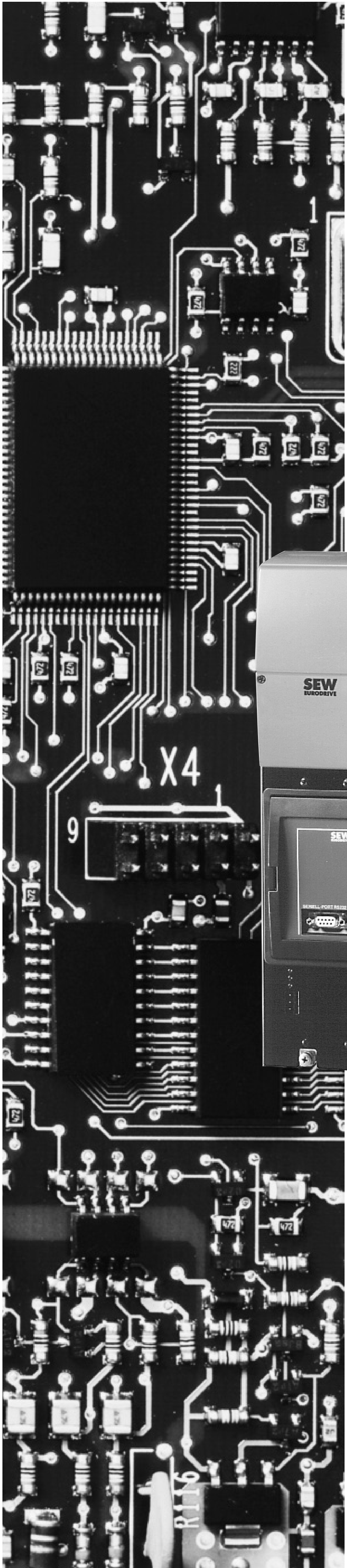


MOVIDYN® Servo Controller

AFP11A ”PROFIBUS Option PCB”

User Manual

Edition 07/96

16/042/95

PROFI PROCESS FIELD BUS BUS

0922 856X / 0898

TÜV CERT DIN EN ISO 9001

SEW EURODRIVE

Preface

This *PROFIBUS (*AFP 11*) Option user manual* describes the procedure for installing the AFP 11 PROFIBUS option pcb in the servo controller and for commissioning the MOVIDYN® Servo Controller when connected to a PROFIBUS-DP or PROFIBUS-FMS fieldbus system.

In addition to describing all the settings on the fieldbus option pcb, this manual further discusses the various options for connecting the servo controller to PROFIBUS-DP or PROFIBUS-FMS in the form of brief commissioning examples.

In addition to this *PROFIBUS Option user manual* the following more detailed documentation on fieldbuses is also necessary in order to enable the MOVIDYN® to be connected simply and efficiently to the PROFIBUS fieldbus system:

- MOVIDYN® Fieldbus Unit Profile user manual
- MOVIDYN® Parameter List

The MOVIDYN®*Fieldbus Unit Profile Manual* gives a detailed description of the fieldbus parameters and their codes and discusses various control concepts and application options in the form of brief commissioning examples.

The MOVIDYN® Parameter List contains a list of all the servo controller parameters that can be read or written via the various communications interfaces such as the RS-232, RS-485, and via the fieldbus interface.

Contents

SEW EURODRIVE

Important Notes

- **Read this user manual carefully before you start installation and commissioning work on MOVIDYN® Servo Controllers with PROFIBUS options.**
  This user manual assumes that the user is familiar with and has at his disposal all relevant documentation on the MOVIDYN® system, in particular the installation and operating instructions.

- **Safety notes:**
  Always follow the safety notes contained in this user manual.
  Safety notes are marked as follows:

Electrical hazard, e.g. during live working

Mechanical hazard, e.g. when working on hoists

Important Instructions for the safe and fault-free operation of the system, e.g. pre-setting before commissioning.
Failure to follow these instructions may result in injury to people and damage to property.

- **General safety notes for bus systems:**
  The fieldbus option gives you a communications system which allows you to match the MOVIDYN® drive system to the specifics of your application to a very high degree. As with all bus systems there is, however, the risk of parameters being changed, which will not show outside (i.e. the servo controller) but affect the behaviour of the servo controller. This may result in unexpected (not uncontrolled, though) system behaviour.

- **In these instructions, cross-references are marked with a , e.g.,**
  (→ MD_SHELL) means: Please refer to the MD_SHELL user manual for detailed information or information on how to carry out this instruction.
  (→ section x.x) means: Further information can be found in section x.x of this user manual.

Each unit is manufactured and tested to current SEW-EURODRIVE technical standards and specifications.

The manufacturer reserves the right to make changes to the technical data and designs as well as the user interface herein described, which are in the interest of technical progress.

A requirement for fault-free operation and fulfilment of any rights to claim under guarantee is that these instructions and notes are followed.

These instructions contain important information for servicing, they should therefore be kept in the vicinity of the unit.

1 Introduction

Thanks to its high-performance, universal fieldbus interface, the MOVIDYN® Servo Controller with the *AFP 11* option enables connections to be made with higher-level automation systems via the open and standardized serial PROFIBUS-FMS and PROFIBUS-DP bus system.

PROFIBUS-FMS

PROFIBUS-FMS (Fieldbus Message Specification) is designed for non-time-critical applications in automation engineering such as, for example, networking different automation systems of various manufacturers. In drive engineering the PROFIBUS-FMS is mainly used for visualization of data and for parameterizing of drives as it allows for larger amounts of non-time-critical data to be exchanged in a simple way. PROFIBUS-FMS is defined in DIN 19245 Part 2.

PROFIBUS-DP

PROFIBUS-DP (Decentralized Periphery) is mainly used for communication with decentralized peripherals, i.e. in the sensor/actuator area, where short system reaction times are required. The main task of PROFIBUS-DP is the fast cyclic data exchange between central automation units (PROFIBUS master) and decentralized peripherals, among them servo controllers. PROFIBUS-DP is defined in DIN E 19245 Part 3.

PROFIBUS-FMS and PROFIBUS-DP can generally be operated on a joint transmission medium. If a joint transmission medium is used, however, the units which are to communicate directly with each other must be able to understand the same protocol option.

MOVIDYN® = Combislave

With the *AFP 11* PROFIBUS option pcb the MOVIDYN® Servo Controller as Combislave unit supports both PROFIBUS-FMS and PROFIBUS-DP. This allows the servo controller to be controlled via PLC and PROFIBUS-DP, for example, while at the same time a visualization system can read out and display on a PC screen actual values from the servo controller using PROFIBUS-FMS. Of course the servo controller may be controlled and parameterized using only PROFIBUS-DP or only PROFI-BUS-FMS, too.

MOVIDYN® and PROFIBUS

The servo controller unit profile for PROFIBUS mode, i.e. the way the servo controller operates and responds when in PROFIBUS mode, is independent of the type of fieldbus, and thus consistent for all fieldbus types. This allows the user to develop his drive applications independent of a particular fieldbus or change to another bus system, e.g. the open standardized INTERBUS-S (A*FI 11* option) sensor/actuator bus.

MOVIDYN® offers digital access to all drive parameters and functions via the PROFIBUS interface. The servo controller is controlled by the high-speed cyclic process data. This process data channel provides the facility to specify setpoints, such as setpoint speeds, ramp generator times for acceleration and deceleration etc., as well as various drive functions such as enable, controller inhibit, stop, rapid stop, etc. to be triggered. This channel can also be used to read back actual values from the servo controller, such as actual speed, current, unit status, error number or reference messages.

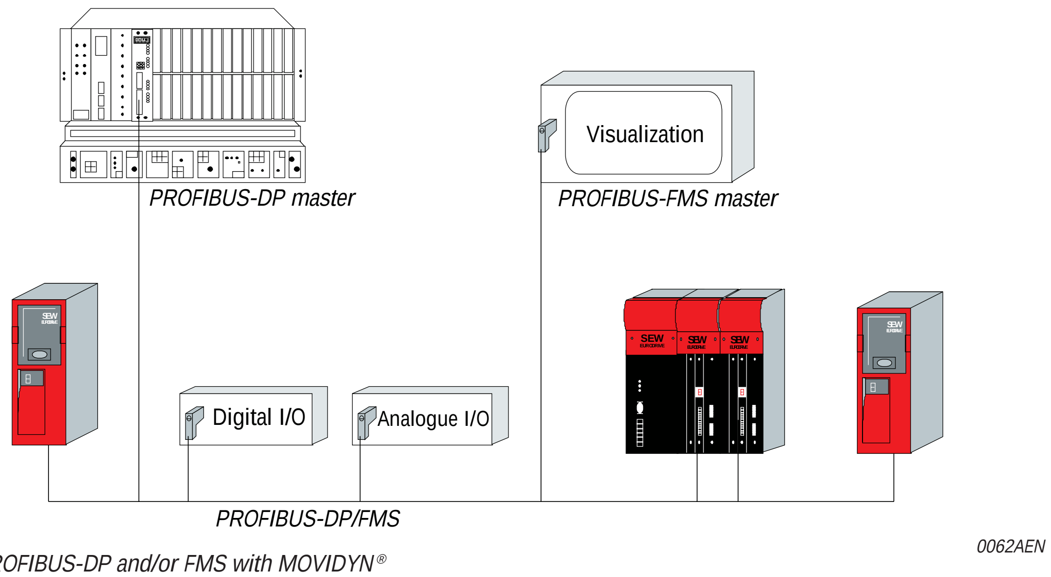


Fig. 1: PROFIBUS-DP and/or FMS with MOVIDYN®

Whereas process data are generally exchanged in cycles, the drive parameters can be read and written acyclically via the READ and WRITE services or the parameter channel. This exchange of parameter data enables applications where all major drive parameters are stored in the higher-level automation unit to be implemented, thus avoiding manual adjustment of parameters on the servo controller itself, which can be very time-consuming.

The PROFIBUS option pcb is designed so that all fieldbus specific settings, such as the station address or the default parameters, can be made on the option pcb by means of a hardware switch. These manual settings enable the servo controller to be integrated into the PROFIBUS environment and switched on in a very short space of time. Parameters can be set fully automatically by the higher-level PROFIBUS master (parameter download). This forward-looking version offers the benefits of a shorter commissioning period for the plant as well as simpler documentation of the application program, as all major drive parameter data can now be recorded directly in the control program.

The use of a fieldbus system in drive technology requires additional monitoring functions, such as fieldbus timeout or special emergency stop concepts. The monitoring functions of the MOVIDYN® can be matched to the specific application for which it is to be used. This feature enables you, for instance, to specify which fault response the servo controller should trigger if an error should occur in the bus. A rapid stop will be practical for many applications, but it is also possible to freeze the last setpoints, so that the drive can continue with the last valid setpoints (e.g. conveyor belt). As the functionality of the control terminals is also ensured when the servo controller is operated in the fieldbus mode, fieldbus-independent emergency stop concepts can still be implemented via the servo controller's terminals.

The MOVIDYN® Servo Controller offers numerous diagnostic facilities for commissioning and servicing. For instance, both the setpoints transmitted from the higher-level control unit as well as the actual values can be checked with the integrated fieldbus monitor. It also provides you with a lot of additional information on the status of the fieldbus option pcb. In combination with the MD_SHELL PC software the fieldbus monitor function offers convenient diagnostic facilities in that it provides a detailed display of the fieldbus and unit status information as well as the facility to set all the drive parameters (including the fieldbus parameters).

2 Assembly / Installation Instructions

Unless the *AFP 11* option is already installed in the MOVIDYN® Servo Controller, please check if the components stated in the scope of delivery are complete.

2.1 Scope of Delivery

The *AFP 11* option comprises the following components:

- 1 *AFP 11* (PROFIBUS) option pcb
- 1 housing cover for MAS... or for MKS...

2.2 Supported Servo Controller Types

The *AFP 11* option pcb for connection to a PROFIBUS-FMS/DP system can be used with all servo controllers of the MOVIDYN® .. 51.. family.

To adjust fieldbus parameters, you need the MD_SHELL PC user interface, ver. V1.60 or higher !

2.3 Fitting the Option PCB

Please follow the instructions below when fitting the option pcb:

Before you start

Option pcbs
Store the option pcb in its original packaging and only take it out shortly before you fit it.
Hold the option pcb by its edge and do not touch unnecessarily. Do not touch any components.

Procedure for Fitting the Option PCB:

1. Disconnect the servo controller from the supply. Switch off the mains supply and, if connected, the external 24V supply.
2. Take off the left black front cover after removing the two recessed head screws.
   Note: When the controller cover is removed, the unit has enclosure IP00. Dangerous voltages may be present for up to 10 minutes after disconnecting the unit from the mains.
3. Take appropriate ESD measures before touching the pcb (wrist strap, conductive shoes, etc.).
4. Position the pcb with the backplane connector to the rear in the guide rails of the option pcb slot. Make sure that the pcb sits properly in the rear guide rails.
5. Press the backplane connector of the pcb into the socket in the controller housing. The pcb sockets must be flush with the cover of the axis module / compact servo controller.
6. Install the supplied cover plate to cover the option pcb slot and screw tight (2 screws).
7. **The *AFP 11* option pcb is now completely fitted.**

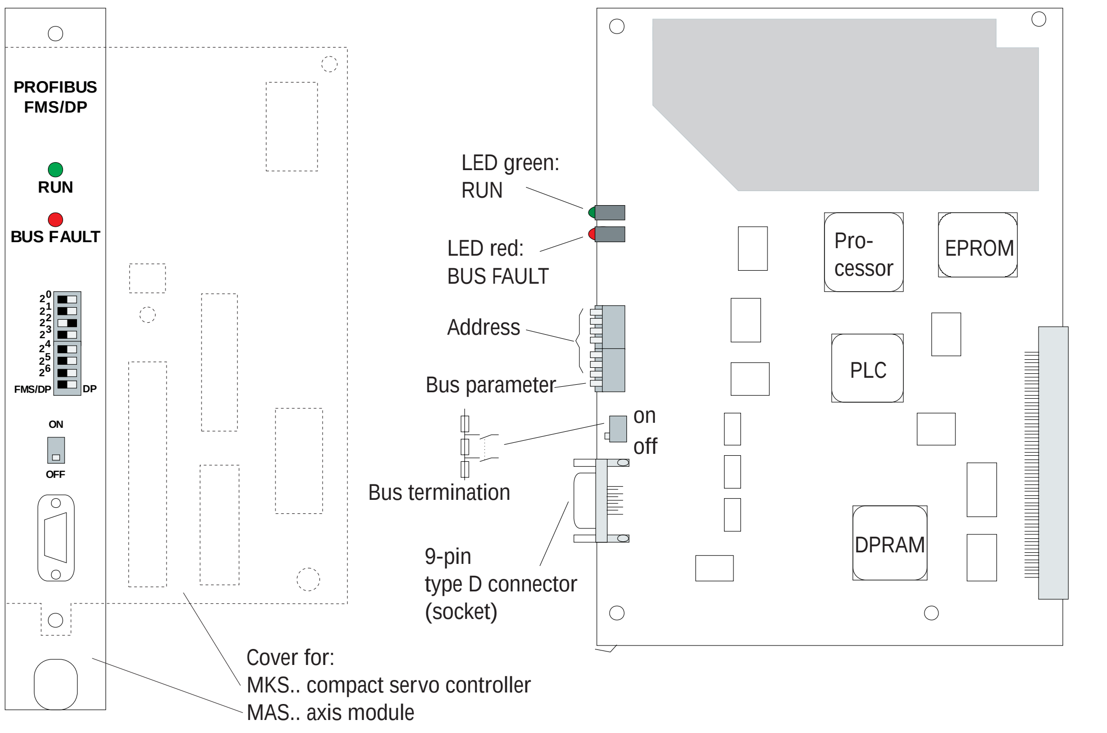
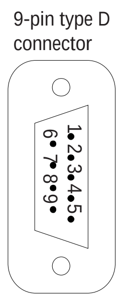


Fig. 2: The AFP 11 option

2.4 Pin Assignment

The MOVIDYN® Servo Controller is connected to the PROFIBUS network via a 9-pin type D connector in accordance with DIN 19245 Part 3. Connection to the T bus is with an appropriately designed connector or a bus terminal. Fig. 3 shows the pin assignment. As the bus terminating resistors can be connected on the option pcb, it is not necessary to use a type D connector with integrated terminating resistors.



Pin no.	Signal		RS-485 reference
1:	-	not assigned	
2:	-	not assigned	
3:	**RxD/TxD-P**	**receive/send data P**	**B/B '**
4:	CNTR-P	repeater control signal (TTL)	
5:	DGND	data reference potential (M 5V)	C/C'
6:	VP	supply voltage plus (P5 V)	
7:	-	not assigned	
8:	**RxD/TxD-N**	**receive/send data N**	**A/A '**
9:	DGND	data reference potential (M 5V)	
Connector housing	Screen of the twisted two-wire cable		

Fig. 3

The MOVIDYN® Servo Controller is connected to the PROFIBUS system via a twisted, screened two-wire cable. The connection of the two-wire cable to the PROFIBUS connector is via pin 8 (A/A') and pin 3 (B/B'). These two contacts are used for communication. The RS-485 signals *A/A'* and *B/B'* must be contacted the same on all PROFIBUS stations, as otherwise communication via the bus will not be possible.

Via pin 4 (CNTR-P) the PROFIBUS option pcb supplies a TTL control signal for a repeater (reference = pin 9).

2.5 Screening and Laying of the Bus Cables

The AFP 11 PROFIBUS option pcb supports RS-485 transmission technology and requires as a physical medium a screened, two-wire twisted-pair cable (cable type A) specified for PROFIBUS in accordance with DIN 19245 Part 3 (see Appendix).

Technically correct screening of the bus cable absorbs the electrical interference that can occur in an industrial environment. You will achieve the best screening results if you adopt the following measures:

- Hand-tighten the fixing screws of plugs, modules and equipotential bonding conductors.·
- Only use plugs with metal or metal-plated housings.·
- Connect the screening in the plug over as large an area as possible.·
- Connect the screening at both ends of the bus cable·
- Do not lay signal and bus cables parallel to power cables (motor cables), but wherever possible in separate cable conduits.·
- In an industrial environment use metallic, earthed cable trays.·
- Run signal cables and the associated equipotential bonding conductor as close as possible to each other, using the shortest route.·
- Avoid extending bus cables through the use of connectors.·
- Run the bus cables close to existing earthed surfaces.

Note:
In the event of fluctuations in the earth potential, a circulating current may flow through any screening which may be connected at both ends and connected to the earth potential (PE). In this case, ensure there is adequate equipotential bonding in accordance with the relevant DIN VDE provisions.

2.6 Bus Termination

If the MOVIDYN® Servo Controller is at the beginning or the end of a PROFIBUS segment, connection to the PROFIBUS network, as a rule, is not via a T bus with an incoming and outgoing PROFIBUS cable but directly with only one PROFIBUS cable. To avoid interferences on the bus system caused by reflections etc., the PROFIBUS segment must be terminated with bus terminating resistors on the physically first and last stations (Fig. 4).

As the bus terminating resistors can be connected on the AFP 11 option pcb of the servo controller it is not necessary to use a type D connector with integrated terminating resistors.

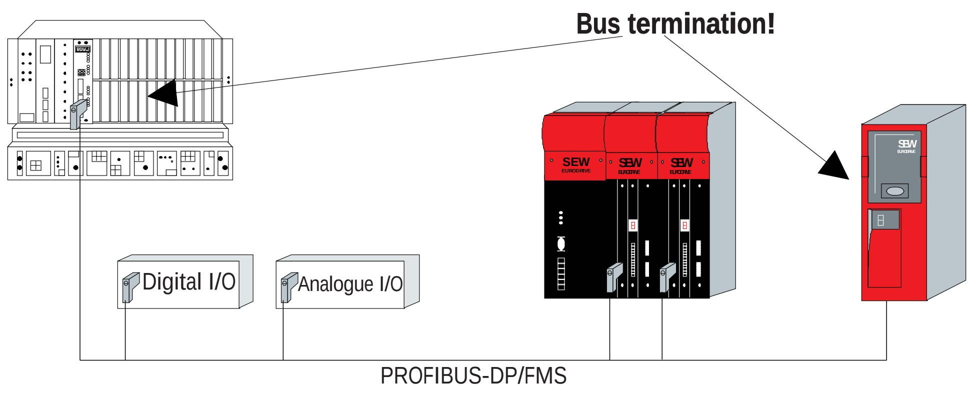


Fig. 4: Bus termination at the beginning and end of a PROFIBUS segment

Set the appropriate DIP switch on the option pcb (see Fig. 5) to the "*on*" position in order to connect the bus terminating resistors (in accordance with DIN 19245 Part 3).

The bus termination for cable type A is implemented in accordance with DIN E 19245 Part 3.

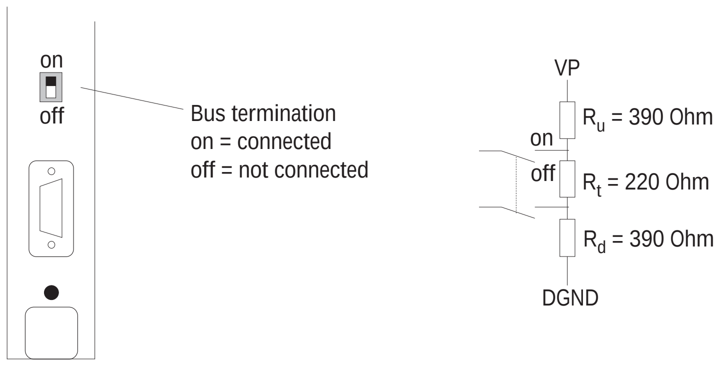


Fig. 5: Activating the bus terminating resistors

2.7 Setting the Station Address

The PROFIBUS station address is set with the DIP switches on the option pcb. PROFIBUS supports the address range from 0 - 125. The address 126 is reserved for PROFIBUS-DP and is for setting the address via the bus interface. This feature is, however, not supported by the MOVIDYN®. The address 127 is reserved for the broadcast service. Fig. 6 shows how the station address is set with the DIP switches.

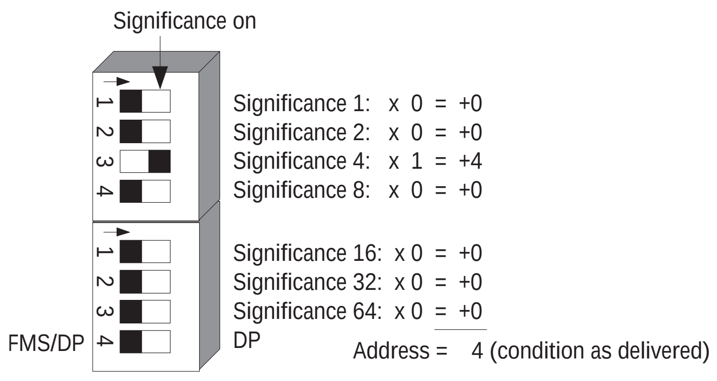
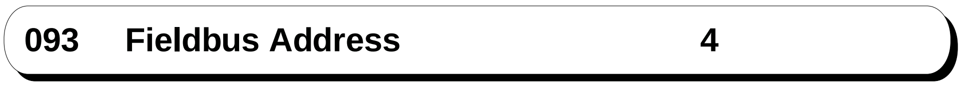


Fig. 6: Setting the PROFIBUS station address

It is not possible to change the PROFIBUS station address via the DIP switches while the servo controller is running. If the station address is changed, the new station address will only be effective after the servo controller has been switched off (mains supply and 24V supply ON/OFF) and then switched on again. The station address set on the servo controller can be displayed in the fieldbus monitor parameter *P073 Fieldbus Address* (see Fig. 7).



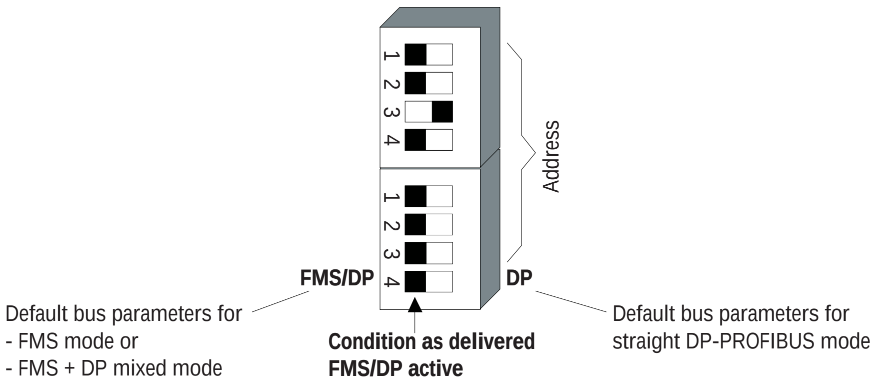
093 Fieldbus Address 4

00084AEN

Fig. 7: Displaying the current PROFIBUS station address

2.8 Setting the Bus Parameters

The default value setting for the bus parameters depends on the protocol option used. For straight PROFIBUS-DP mode the DIP switch must be set at DP. This will activate the default bus parameters (in particular the min T_{SDR}) for time-optimized DP mode in accordance with DIN E 19245 Part 3. For mixed mode (FMS/DP) or straight FMS mode the DIP switch must be set at FMS (Fig. 8).



00083AEN

Fig. 8: Setting the default bus parameters to DIN 19245

This switch only serves to select the default bus parameters. Independent of the setting of this switch the servo controller at any time supports simultaneous use of the PROFIBUS protocol options FMS and DP (Combislave functionality).

Any change to this DIP switch setting will only become effective after the servo controller has been switched off (mains supply and 24V supply ON/OFF) and switched on again.

2.9 Display Elements

LED Green *RUN*	LED Red *BUS ERROR*	Meaning
Flashing at approx. 3Hz	Off	Option pcb is being initialized (only immediately after servo controller power-up or reset)
On	Flashing at approx. 0.5Hz	Configured station address is not within the permitted range (0...126) ⇒Set correct station address and switch the unit on again.
On	Irrelevant	Normal operation - AFP 11 option pcb is working correctly

On	On	<u>DP mode:</u> a) When the servo controller is being commissioned or accelerating: The servo controller has not been set to data exchange mode by the DP master yet . b) The timeout period has elapsed, the servo controller was not addressed by the DP master. <u>FMS mode:</u> There is no active FMS link between the FMS master and the servo controller. <u>Mixed mode FMS/DP:</u> A combination of the above
On	Off	<u>DP mode:</u> The servo controller is in data exchange mode. <u>FMS mode:</u> There is an active FMS link between the FMS master and the servo controller. <u>Mixed mode FMS/DP:</u> Combination of the above
On+ flickering	Off	The servo controller is being parameterized via PROFIBUS-DP or -FMS (Read/Write access operations)
Off	On	Hardware fault on the AFP 11 option pcb
Flashing at approx. 1Hz	Flashing at approx. 1Hz	Hardware fault on the AFP 11 option pcb
Flashing at approx. 1 Hz	On	Check the firmware version of the control pcb (basic unit).

Table 1: Meaning of the visual signals of the RUN and BUS ERROR LEDs

The option pcb has two LEDs for status and fault indication of the option pcb and the connected bus system (Fig. 2). Table 1 shows the meaning of the visual signals of these LEDs. While the green LED "*RUN*" indicates the operational status of the option pcb, the red LED "*BUS ERROR*" indicates the status of the PROFIBUS connection.

2.10 Commissioning the Servo Controller

After installing the PROFIBUS option pcb the MOVIDYN® Servo Controller can be immediately parameterized via the PROFIBUS system without any further manual intervention. This means, for example, that after switching on the servo controller, all parameters can be downloaded directly from the higher-level control.

To control the servo controller via PROFIBUS, however, it must first be switched to the appropriate setpoint source. This is possible using the parameter *P110 Setpoint Source = FIELDBUS*. The factory setting for this parameter is *ANALOGUE INPUT*. Using the parameter *FIELDBUS,* the servo controller is programmed to accept setpoints from the PROFIBUS. MOVIDYN® now responds to process data sent from the higher-level control.

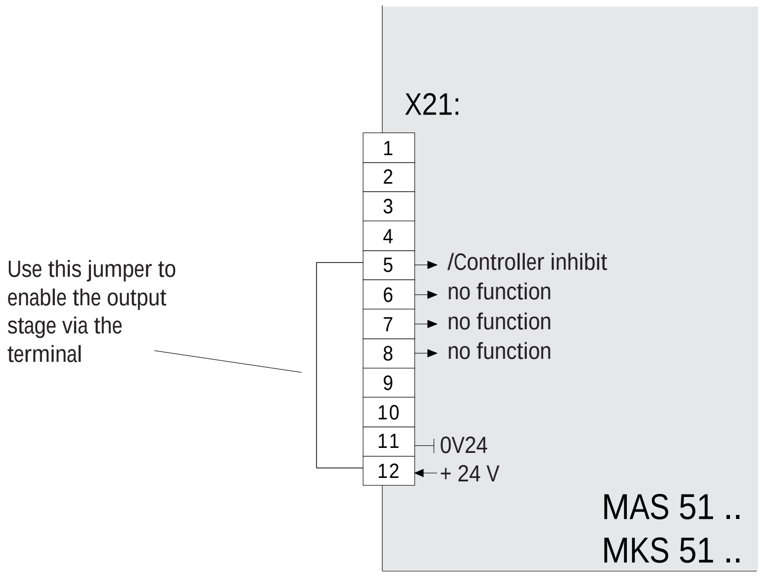
The activation of the FIELDBUS setpoint source mode is signalled to the higher-level control by the *Fieldbus Mode Active* bit in the status word.

For safety reasons the servo controller must also be enabled on the terminal side to permit control via the fieldbus system. The terminals are therefore to be wired or programmed in such a way that the servo controller is enabled via the input terminals. The easiest way of enabling the servo controller on the terminal side is, for example, to connect input terminal X21.5 (*/CONTROLLER INHIBIT* function) to a +24V signal and program input terminals X21.6-8 to *NO FUNCTION*. An example of the commissioning procedure for the MOVIDYN® Servo Controller with a fieldbus interface is given on the following page.

Commissioning procedure for the MOVIDYN® Servo Controller

1. Enable the output stage on the terminal side.

Apply a +24V signal on input terminal 21.5 (/CONTROLLER INHIBIT function) (e.g. via jumper)



00085AEN

Fig. 9: Enabling the output stage via jumper

2. Switch on 24 V supply

Switch on the external 24V supply only (not the mains supply !) to reprogram the servo controller to setpoint source “FIELDBUS” despite the installed jumper.

3. Input terminals X21.6 … 8 = NO FUNCTION

Program input terminals X21.6, X21.7 and X21.8 to NO FUNCTION. In “Positioning” mode you may continue to use terminals X21.7 and X21.8 as limit switches.

P300	Programming terminal MA (X21.6) = NO FUNCTION
P301	Programming terminal MA (X21.7) = NO FUNCTION (unless in positioning mode)
P302	Programming terminal MA (X21.8) = NO FUNCTION (unless in positioning mode)

4. Setpoint source = FIELDBUS

Set the setpoint source to FIELDBUS to control the servo controller via fieldbus.

P110	Programming terminal MA (X21.6) = NO FUNCTION

For more information on commissioning and controlling the MOVIDYN® Servo Controller please refer to the *Fieldbus Unit Profile User Manual* documentation.

3 The PROFIBUS-DP Interface

PROFIBUS-DP (Decentralized Periphery) is the speed-optimized PROFIBUS option designed in particular for fast data exchange at the sensor/actuator level. DP is used by central automation systems (e.g. programmable logic controllers) to communicate with decentralized peripherals such as sensors and actuators, among them servo controllers, via a fast serial link. Data exchange with these decentralized units is mainly cyclic. The central control system sends new process output data to all slaves in a message and reads in all process input data from the slaves (sensors, actuators) in the same message.

The considerable increase in speed of PROFIBUS-DP compared to PROFIBUS-FMS is primarily due to the fact that DP has no application layer (layer 7) and I/O data are transferred between the master and a slave in a single message cycle. A maximum of 246 bytes of I/O data can be transferred between the DP master and a DP slave. Normally, however, shorter data blocks (of up to 32 bytes) are used to increase efficiency still further. Consequently, the exchange of data via PROFIBUS-DP can be seen as a straight process communications procedure.

To enable the DP master to communicate with the DP slaves, it has to be given some important information regarding the DP interface of the connected slave. In addition to data relating to the type and amount of I/O data to be transferred, it also requires additional information regarding the identity of each DP slave.

3.1 Configuration of the DP Interface

To be able to define the type and amount of I/O data to be transferred, the DP master has to pass a certain configuration to the servo controller. The MOVIDYN® Servo Controller can generally be operated using six different configurations. You have the option of only controlling the servo controller by exchanging process data, or, in addition to controlling the servo controller via process data, of reading or writing parameters using an additional parameter channel at the same time.

Fig. 10 provides a schematic representation of the exchange of data between the programmable automation unit (DP master) and the MOVIDYN® Servo Controller (DP slave) using process data and parameter channels.

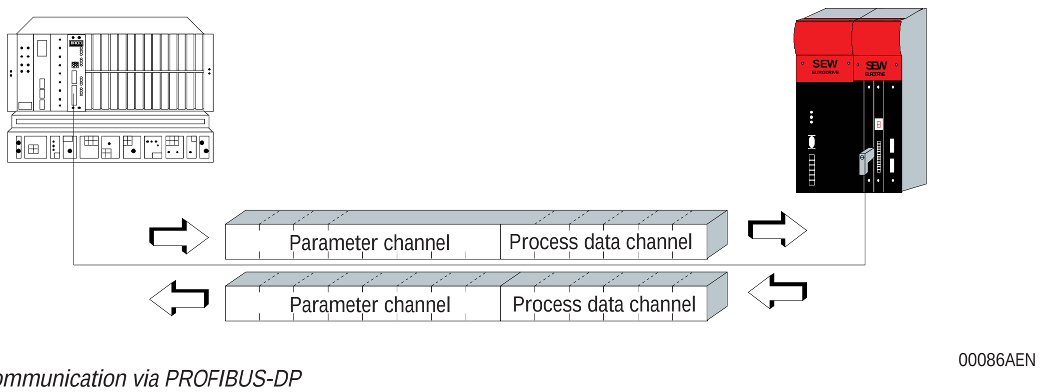


Fig. 10: Communication via PROFIBUS-DP

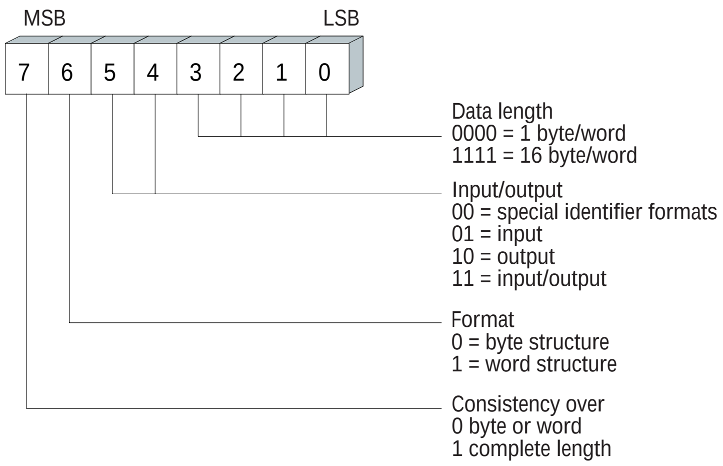
When commissioning the DP master, you will have to specify which configuration is going to be used to operate the servo controller. This configuration is then transferred to the servo controller when the DP master is started up (using the DDLM_Chk_Cfg service). The servo controller checks the transferred configuration data for plausibility before going into data exchange mode.

The configuration data are coded in accordance with DIN E 19245 Part 3 and are discussed in the next section.

3.1.1 Description of the Configuration Data

DIN E 19245 Part 3 describes the format of the configuration data. Fig. 11 shows the Cfg_Data identifier byte which, according to DIN E 19245 Part 3, is used to describe which I/O data are to be transferred between master and slave using PROFIBUS-DP.

In addition to specifying the data length in bits 0-3, you have to use bits 4 and 5 to define whether the transfer involves input and/or output data. Bit 6 indicates whether the data are to be transferred in byte or word format and bit 7 is used to specify the consistency with which the data are to be handled by the bus system. For example, position values of the MOVIDYN® Servo Controller should be transferred in a consistent manner, i.e. it has to be ensured that contiguous data are also transferred together and not, for example, that the least significant part of the position is transferred one bus cycle ahead of the more significant part.



00087AEN

Fig. 11: Format of the Cfg_Data identifier byte to DIN E 19245 Part 3

The MOVIDYN® Servo Controller supports six different process data configurations. To control the servo controller, you can define the amount of process data to be transferred using 1, 2 or 3 process data words and also enable/disable a parameter channel for read/write access to all drive parameters. This produces the following process data configurations:

1 process data word (1 PD)
2 process data words (2 PD)
3 process data words (3 PD)
1 process data word + parameter channel (1 PD + Param)
2 process data words + parameter channel (2 PD + Param)
3 process data words + parameter channel (3 PD + Param)

This configuration is set up solely via the DP master as the bus system is started up, so that no additional manual parameterizing of the servo controller is required. This automatic configuring mechanism enables download applications to be implemented where the servo controller can be completely controlled and parameterized via the bus system.

To set these process data configurations, the servo controller supports a number of different codes for the Cfg_Data identifier byte. The process data configuration is allocated based on the amount of input and output data.

A valid DP configuration sent from the DP master to the servo controller must conform to the following conventions:

- The amount of input or output data must correspond to the contents of Table 2.
- The number of input bytes and output bytes must be the same.

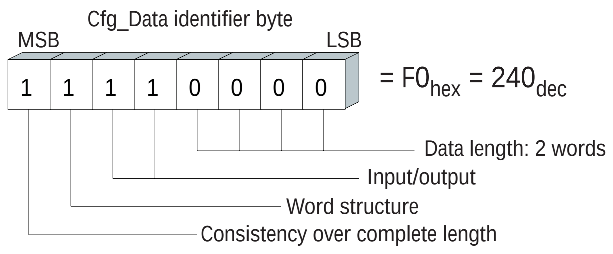
Length of the input/output data	Meaning
2 bytes or 1 word	1 process data word
4 bytes or 2 words	2 process data words
6 bytes or 3 words	3 process data words
10 bytes or 5 words	1 process data words + parameter channel
12 bytes or 6 words	2 process data words + parameter channel
14 bytes or 7 words	3 process data words + parameter channel

Table 2: Possible data lengths of the DP configuration and their interpretation

The servo controller interprets the length of the DP configuration passed to it as shown in Table 2. The six different process data configurations for PROFIBUS-DP are described below.

3.1.2 Configuring for 1 Process Data Word (1 PD)

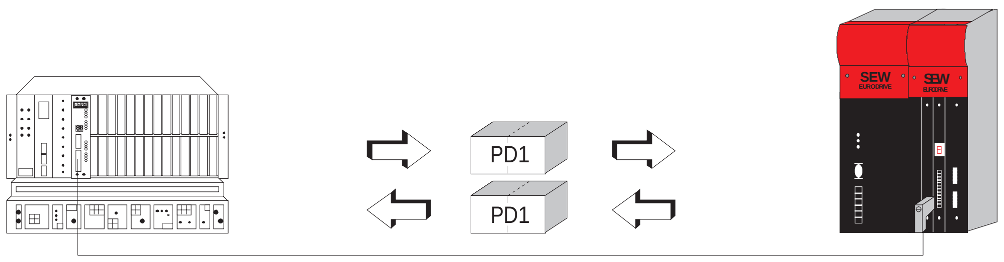
Control of the MOVIDYN® Servo Controller using only one process data word requires, for example, that the Cfg_Data identifier byte is coded as shown in Fig. 12. This code must be sent to the servo controller by the DP master when PROFIBUS-DP is started so that the DP master and DP slave can exchange a process data word.



00088AEN

Fig. 12: Configuration data example for setting 1 input/output word (1 PD)

Fig. 13 shows the communication between the automation unit (DP master) and the MOVIDYN® Servo Controller via one process data word only. This configuration could be used, for example, to control the servo controller with control word 1 and status word 1 (see SEW documentation *Fieldbus Unit Profile User Manual*).

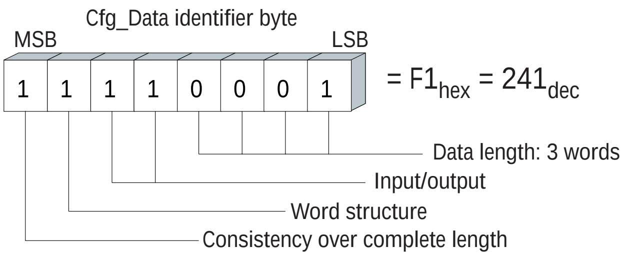


00089AXX

Fig. 13: Control of the servo controller via 1 process data word

3.1.3 Configuring for 2 Process Data Words (2 PD)

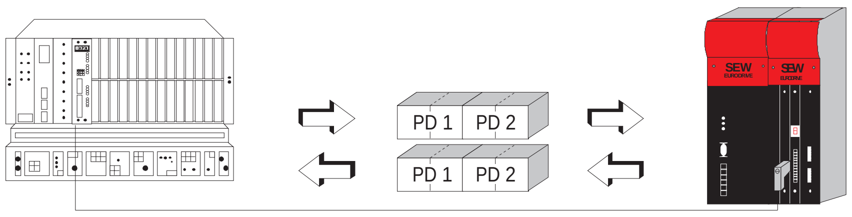
Control of the MOVIDYN® Servo Controller using two process data words requires that the Cfg_Data identifier byte is coded as shown in Fig. 14. This code must be sent to the servo controller by the DP master when PROFIBUS-DP is started so that the DP master and DP slave can exchange two process data words.



00090AEN

Fig. 14: Configuration data example for setting 2 input/output words (2PD)

Fig. 15 shows the communication between the automation unit (DP master) and the MOVIDYN® Servo Controller via two process data words. The higher-level control system could use this configuration, for example, to send the process output data *Control Word 1* and *Speed Setpoint* to the servo controller and read in the process input data *Status Word 1* and *Speed Actual Value* (see SEW documentation *Fieldbus Unit Profile User Manual*).

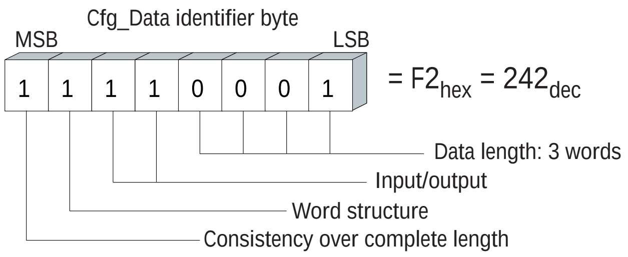


00091AXX

Fig. 15: Control of the servo controller via 2 process data words

3.1.4 Configuring for 3 Process Data Words (3 PD)

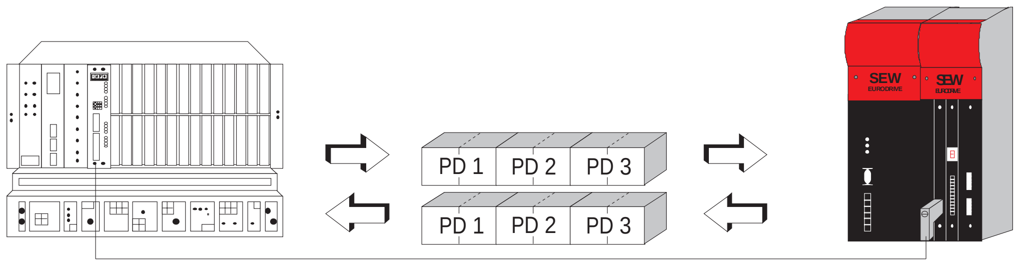
Control of the MOVIDYN® Servo Controller using three process data words requires that the Cfg_Data identifier byte is coded as shown in Fig. 16. This code must be sent to the servo controller by the DP master when PROFIBUS-DP is started so that the DP master and DP slave can exchange three process data words.



00092AEN

Fig. 16: Configuration data example for setting 3 input/output words (3PD)

Fig. 17 shows the communication between the automation unit (DP master) and the MOVIDYN® Servo Controller via three process data words. The higher-level control system could use this configuration, for example, to send the process output data *Control Word 1*, *Speed Setpoint, Process Ramp* and read in the process input data *Status Word 1*, *Speed Actual Value* and *Apparent Current Actual Value* (see SEW documentation *Fieldbus Unit Profile User Manual*).

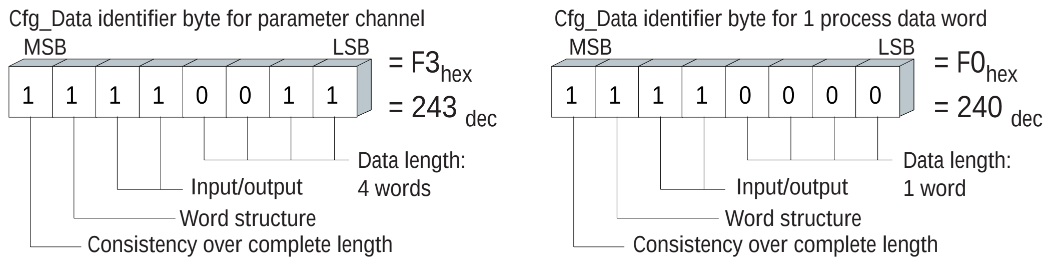


00093AXX

Fig. 17: Control of the servo controller via 3 process data words

3.1.5 Configuring for 1 PD + Parameter Channel

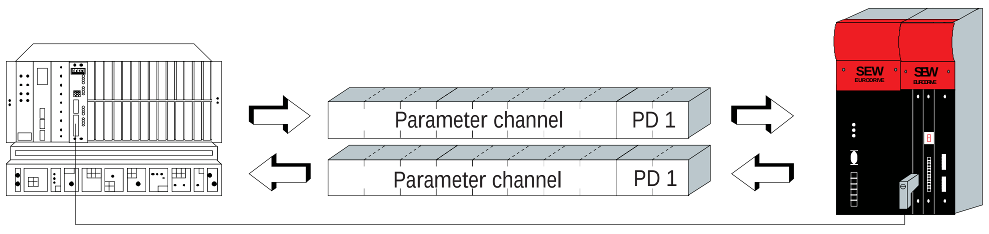
Control of the MOVIDYN® Servo Controller using one process data word and an additional parameter channel requires two identifier bytes to be defined. Identifier byte 1 contains the code for the parameter channel, identifier byte 2 contains the code for a single process data word. Fig. 18 shows how these two identifier bytes are coded. These codes must be sent to the servo controller by the DP master when PROFIBUS-DP is started so that the DP master and the DP slave can exchange the process data word as well as the parameter channel.



00094AEN

Fig. 18: Configuration example for parameter channel + 1 process data word

Fig. 19 shows the communication between the automation unit (DP master) and the MOVIDYN® Servo Controller via one process data word and the parameter channel for reading and writing of drive parameters. The higher-level control system could use this configuration, for example, to control the servo controller with *Control Word 1* and *Status Word 1* and access all drive parameters via the parameter channel (see SEW documentation *Fieldbus Unit Profile User Manual*).



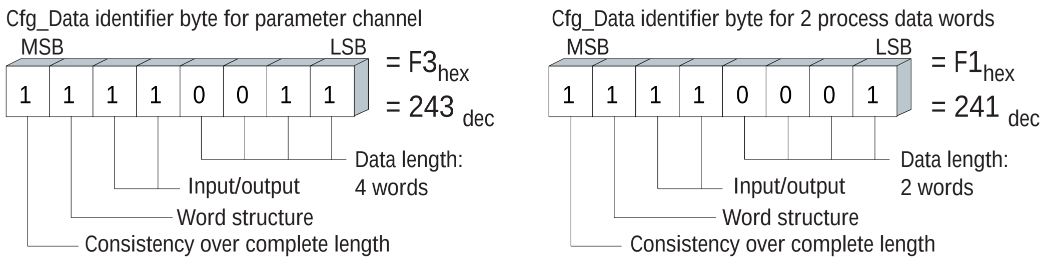
00095AEN

Fig. 19: Communication with 1 process data word and parameter channel

3.1.6 Configuring for 2 PD + Parameter Channel

Control of the MOVIDYN® Servo Controller using two process data words and an additional parameter channel requires two identifier bytes to be defined. Identifier byte 1 contains the code for the parameter channel, identifier byte 2 contains the code for two process data words. Fig. 20 shows how these two identifier bytes are coded.

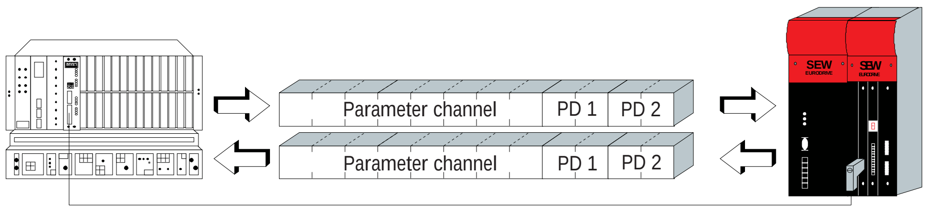
These codes must be sent to the servo controller by the DP master when PROFIBUS-DP is started so that the DP master and the DP slave can exchange two process data words as well as the parameter channel.



00096AEN

Fig. 20: Configuration example for parameter channel and 2 process data words

Fig. 21 shows the communication between the automation unit (DP master) and the MOVIDYN® Servo Controller via two process data words and the parameter channel for reading and writing of drive parameters. This configuration could be used, for example, to control the servo controller with *Control Word 1, Speed Setpoint* and *Status Word 1, Speed Actual Value* resp. and parameterize it via the parameter channel (see SEW documentation *Fieldbus Unit Profile User Manual*).

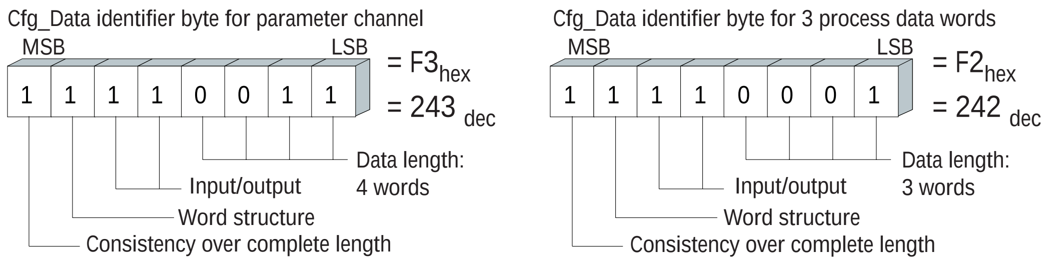


00097AEN

Fig. 21: Communication with 2 process data words and parameter channel

3.1.7 Configuring for 3 PD + Parameter Channel

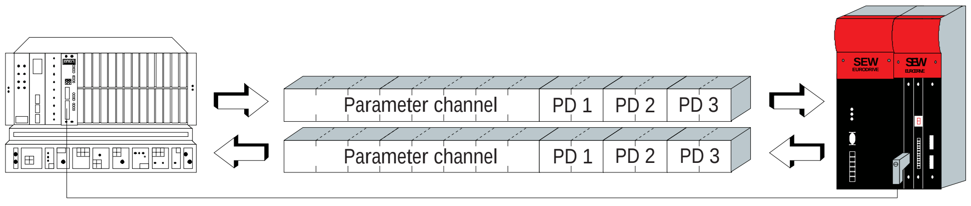
Control of the MOVIDYN® Servo Controller using three process data words and an additional parameter channel requires two identifier bytes to be defined. Identifier byte 1 contains the code for the parameter channel, identifier byte 2 contains the code for three process data words. Fig. 22 shows how these two identifier bytes are coded. These codes must be sent to the servo controller by the DP master when PROFIBUS-DP is started so that the DP master and the DP slave can exchange three process data words as well as the parameter channel.



00098AEN

Fig. 22: Configuration example for parameter channel + 3 process data words

Fig. 23 shows the communication between the automation unit (DP master) and the MOVIDYN® Servo Controller via three process data words and the parameter channel for reading and writing of drive parameters. This configuration could be used, for example, to control the servo controller with *Control Word 1, Speed Setpoint, Process Ramp* and *Status Word 1, Speed Actual Value, Apparent Current Actual Value* resp. and parameterize it via the parameter channel (see SEW documentation *Fieldbus Unit Profile User Manual*).

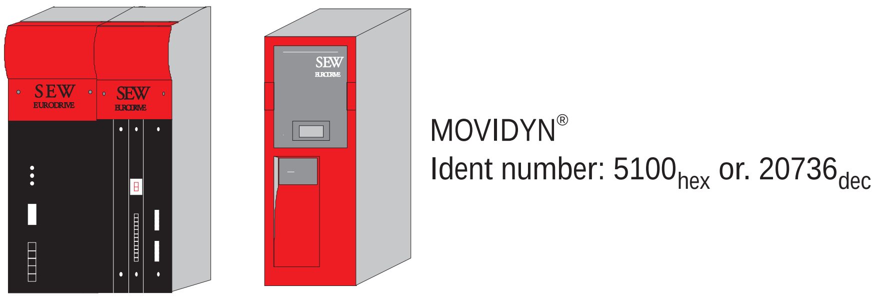


00099AEN

Fig. 23: Communication with 3 process data words and parameter channel

3.2 Ident Number

Each DP master and DP slave must have an individual identification number assigned to them by the PROFIBUS User Group so that the units connected to the bus can be uniquely identified. When the PROFIBUS-DP master is started, it compares the Ident Numbers of the connected DP units with those specified by the user. User data transfer is activated once the DP master has ascertained that the connected station addresses and unit types (Ident Numbers) agree with those specified. This process provides a high degree of security against configuration errors.



00134AEN

Fig. 24: Ident Number of the MOVIDYN® family

The Ident Number is defined as an unsigned 16-bit number (Unsigned16). The PROFIBUS User Group has specified the Ident Number 5100$_{hex}$ (20736$_{dec}$) for the MOVIDYN® range of servo controllers (Fig. 24).

3.3 Watchdog Timer

Each DP slave must have a watchdog timer so it can detect a failure of the DP master or the communications link. If no data are transferred between the DP master and DP slave within the specified timeout period, the slave must automatically switch its outputs to a safe state.

The MOVIDYN® Servo Controller maps the timeout period which is defined when the DP master is configured to parameter P791 *Fieldbus Timeout* (Fig. 25). This parameter consequently reflects the currently configured timeout period. If the watchdog timer is not active, the parameter will have a value of *650.00* seconds.

791 FIELDBUS TIMEOUT [s] 0.20

00135AEN

Fig. 25: A fieldbus timeout period of 200ms configured in the DP master

When the timeout period expires, the servo controller invokes the fault response specified beforehand in parameter P79*2 Timeout Response*. This means the response of the servo controller when the bus goes down can be adapted to that of the drive application. For example, conveyor belts can continue to run at the most recent valid setpoint speed or brought to a stop very quickly.

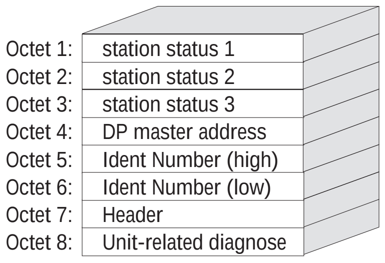
Expiry of the timeout period is indicated on the option pcb by the red *BUS ERROR* LED. At the same time, the servo controller also indicates an error in the 7-segment display, which is displayed as *Error Fieldbus Timeout* in the MD_SHELL user interface on your PC (Fig. 25). Depending on the specified fault response, the servo controller may have to be reset to restore its normal status.

For a detailed description of the servo controller's timeout behaviour, please refer to the *Fieldbus Unit Profile User Manual*.

Note:
Parameter P79*1 Fieldbus Timeout* can only be set through the timeout period which is configured in the DP master for the whole DP system. Manual setting of this parameter with the MD_SHELL user interface has no effect, any setting would be overwritten when PROFIBUS-DP is started up the next time.

3.4 Diagnostic Data

Station diagnosis of the MOVIDYN® Servo Controller can be performed using the DP service DDLM_Slave_Diag. The servo controller also supports unit-related diagnosis. Fig. 27 shows the structure of the diagnostic data.


Octet 1: station status 1
Octet 2: station status 2
Octet 3: station status 3
Octet 4: DP master address
Octet 5: Ident Number (high)
Octet 6: Ident Number (low)
Octet 7: Header
Octet 8: Unit-related diagnose

00136AEN

Fig. 26: Structure of the diagnostic data for the MOVIDYN®

Octets 1-7 contain the diagnostic information according to DIN E 19245 Part 3. As the header of the unit-related diagnostic data, a value of 2 in octet 7 indicates that the unit-related diagnostics are 2 bytes in length (incl. header). If there is a fault on the servo controller, octet 8 will also contain the fault code (only then will external diagnosis be possible).

Note
The unit-related diagnostic information is only updated every 800ms. This means that an error message may not be output for 800ms after the fault occurs. A much faster and simpler method of detecting faults can be implemented using status word 1 of the MOVIDYN® Servo Controller (see Fieldbus Unit Profile User Manual).

3.4.1 Data in Octet 1: Station Status 1

Fig. 27 shows the coding of octet *Station Status 1* in accordance with DIN E 19245 Part 3. Station status 1 comprises information which is either generated by the master or by the DP slave itself.

The bits which are controlled by the master are generally set to zero by the DP slave. In the following the individual status bits will be discussed in greater detail.

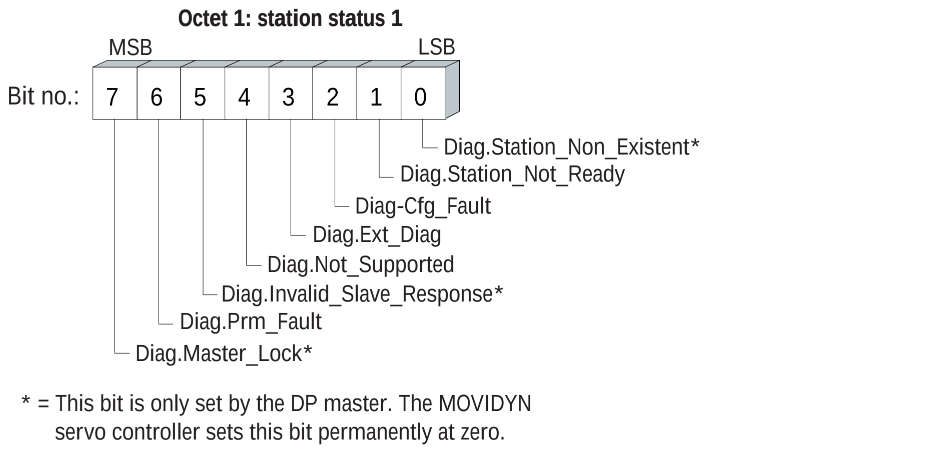

Octet 1: station status 1
MSB
LSB
Bit no.:
7
6
5
4
3
2
1
0
Diag.Station_Non_Existent*
Diag.Station_Not_Ready
Diag-Cfg_Fault
Diag.Ext_Diag
Diag.Not_Supported
Diag.Invalid_Slave_Response*
Diag.Prm_Fault
Diag.Master_Lock*
* = This bit is only set by the DP master. The MOVIDYN servo controller sets this bit permanently at zero.
00137AEN

Fig. 27: Coding of the octet Station Status 1 to DIN E 19245 Part 3

The individual bits have the following meaning in accordance with DIN E 19245 Part 3:

Bit 7: Diag.Master_Lock

The servo controller as DP slave sets this bit permanently at zero.
This bit is set by the DP master (class 1) if the address in octet 4 is not equal FF_{hex} and not equal to its own address. It indicates that the MOVIDYN® Servo Controller was parameterized by a different master with the *DDLM Set_Prm* service.

Bit 6: Diag.Prm_Fault

This bit is set by the MOVIDYN® Servo Controller as DP slave if the last parameter message (*DDLM_Set_Prm*) was incorrect, e.g. incorrect length, incorrect Ident Number, etc.

Bit 5: Diag.Invalid_Slave_Response

The servo controller as DP slave sets this bit permanently at zero.
This bit is set by the DP master if an invalid response was received from the MOVIDYN® Servo Controller.

Bit 4: Diag.Not_Supported

This bit is set by the MOVIDYN® Servo Controller as DP slave if a function was requested which is not supported by the servo controller.

Bit 3: Diag.Ext_Diag

This bit is set by the MOVIDYN® Servo Controller as DP slave.
It indicates that a diagnostic entry has been made in the unit-related diagnosis section (see octet 8: *Unit-related Diagnosis*).

Bit 2: Diag.Cfg_Fault

This bit is set by the MOVIDYN® Servo Controller as DP slave if the configuration data last received by the master do not correspond to the configuration data supported by the MOVIDYN® Servo Controller.

Bit 1: Diag.Station_Not_Ready

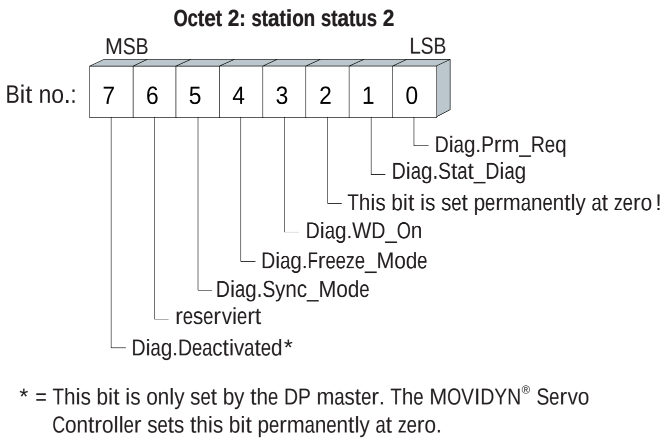
This bit is set by the MOVIDYN® Servo Controller as DP slave if the servo controller is not ready for data exchange yet.

Bit 0: Diag.Station_Non_Existent

The servo controller as DP slave sets this bit permanently at zero. This bit is set by the DP master if the MOVIDYN® Servo Controller cannot be accessed via the bus. If this bit is set, the diagnostic bits in the master contain the status of the last diagnostic message of the servo controller or the initial value resp.

3.4.2 Data in Octet 2: Station Status 2

Fig. 29 shows the coding of octet *Station Status 2* in accordance with DIN E 19245 Part 3. Station status 2 comprises information which is generated either by the master or the DP slave itself. The bits which are controlled by the master are generally set to zero by the DP slave. In the following the individual status bits will be discussed in greater detail.


Octet 2: station status 2
MSB
LSB
Bit no.:
7
6
5
4
3
2
1
0
Diag.Prm_Req
Diag.Stat_Diag
This bit is set permanently at zero !
Diag.WD_On
Diag.Freeze_Mode
Diag.Sync_Mode
reserviert
Diag.Deactivated*
* = This bit is only set by the DP master. The MOVIDYN® Servo Controller sets this bit permanently at zero.

00138AEN

Fig. 28: Coding of the octet Station Status 2 to DIN E 19245 Part 3

The individual bits have the following meaning in accordance with DIN E 19245 Part 3:

Bit 7: Diag.Deactivated

The servo controller as DP slave sets this bit permanently at zero.
This bit is set by the DP master if the MOVIDYN® Servo Controller was identified as non-active in the DP slave parameter set and taken off the cyclic processing.

Bit 6: Reserved

Bit 5: Diag.Sync_Mode

This bit is set by the servo controller as soon as it has received the Sync command.

Bit 4: Diag.Freeze_Mode

This bit is set by the servo controller as soon as it has received the Freeze command.

Bit 3: Diag.WD_On

This bit is set by the MOVIDYN® Servo Controller if the watchdog timer is on.

Bit 2:

This bit is permanently set to one by the MOVIDYN® Servo Controller.

Bit 1: Diag.Stat_Diag

If the MOVIDYN® Servo Controller sets this bit, the DP master must pick up diagnostic data until this bit is cleared again.

Bit 0: Diag.Prm_Req

This bit is set by the MOVIDYN® Servo Controller if it needs to be parameterized and configured again. This bit remains set until the servo controller has been parameterized with DDLM_Set_Prm.

3.4.3 Data in Octet 3: Station Status 3

Fig. 29 shows the coding of octet *Station Status 3* in accordance with DIN E 19245 Part 3. In station status 3 presently only bit 7 is relevant at the moment. Bits 0-6 are reserved.

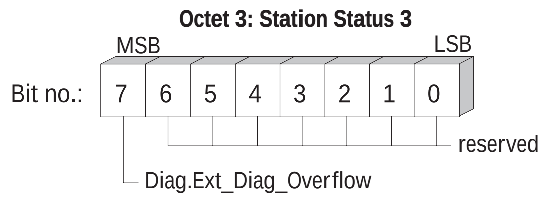


Fig. 29: Coding of the octet Station Status 3 to DIN E 19245 Part 3

The individual bits have the following meaning in accordance with DIN E 19245 Part 3:

Bit 7: Diag.Ext_Diag_Overflow

If this bit is set more diagnostic information is present than specified in Ext_Diag_Data. This bit is generally set to zero by the MOVIDYN® Servo Controller.

Bits 6-0: Reserved

3.4.4 DP Master Address in Octet 4

In this octet the address of the DP master is entered, by which the MOVIDYN® Servo Controller was parameterized with the DP service *DDLM_Set_Prm*. If the servo controller was not parameterized by a DP master, this octet contains the address FF_{hex}.

3.4.5 Ident Number in Octet 5/6

The manufacturer identification for the DP slave type is allocated by the PROFIBUS User Group. This Ident Number can be used both for checking purposes and for exact unit identification. The MOVIDYN® Servo Controller enters the Ident Number 5100_{hex} in this octet, the more significant part (51_{hex}) is entered in octet 5, the less significant part (00_{hex}) is entered in octet 6 of the Ident Number.

3.4.6 Unit-related Diagnosis using Octet 7/8

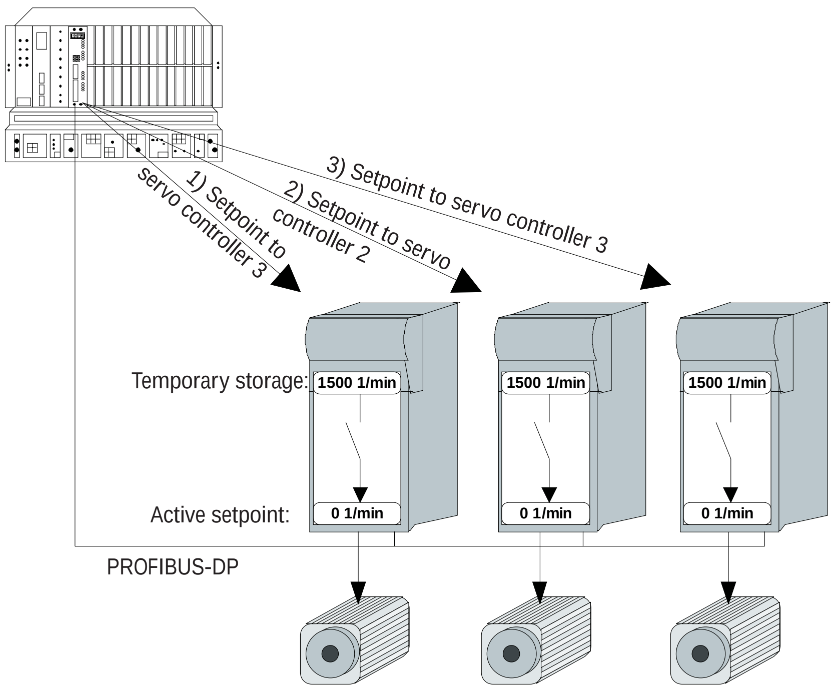
The MOVIDYN® Servo Controller supports unit-related diagnosis. Unit-related diagnostic information is only available when the servo controller outputs a fault message or warning. By setting bit 3 *Diag.Ext_Diag* in octet 1 S*tation Status 1*, the servo controller indicates to the master that unit-related diagnostic information is available.

The unit-related diagnostic information is stored in octet 7 and more specifically in octet 8. As the header of the unit-related diagnostic data, octet 7 contains the length of the unit-related diagnostics (incl. header byte). As the fault code is normally returned as external diagnostic information in one byte, octet 7 normally contains the value 02_{hex} (length 2 bytes) and octet 8 contains the error code from the servo controller, the codes for which can be found in the *MOVIDYN® Parameter List* documentation.

3.5 Sync and Freeze Mode

In addition to the cyclic exchange of data, where the DP master addresses all slave stations in turn, the DP master also has the ability to send various control commands to all slaves or just a group of slaves (multicast functions). These control commands permit event-driven synchronization of the DP slaves.

The *Sync* control command switches the servo controllers into *Sync Mode*. The active setpoint values are frozen when in this mode. The DP master now has enough time to send the new process output data (setpoints) to those stations currently in Sync mode (Fig. 31).


1) Setpoint to servo controller 3
2) Setpoint to servo controller 2
3) Setpoint to servo controller 3
Temporary storage:
1500 1/min
1500 1/min
1500 1/min
Active setpoint:
0 1/min
0 1/min
0 1/min
PROFIBUS-DP

00140AEN

Fig. 30: Sending setpoints to the servo controllers and saving them temporarily

When a new Sync command is issued, all servo controllers simultaneously update their active setpoints with the value temporarily stored beforehand (Fig. 31). In other words, the active setpoints are not updated until the new Sync command has been received. The servo controllers quit Sync mode when the *Unsync* control command is issued.

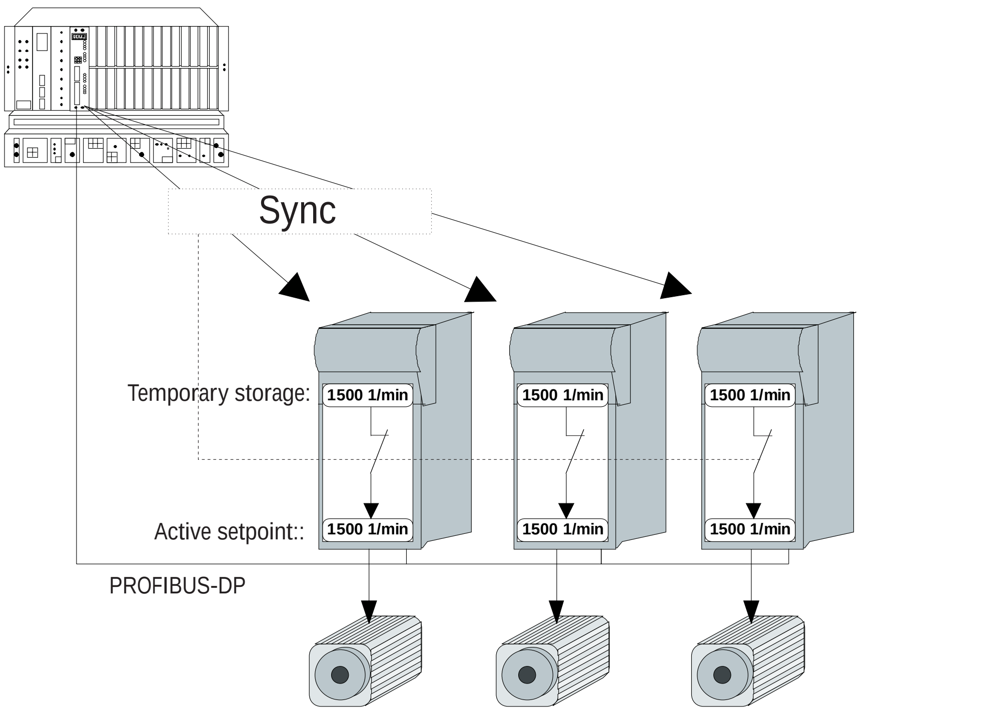


Fig. 31: Simultaneous activation of the new setpoints with the Sync command

The *Freeze* control command switches the addressed slaves into *Freeze Mode*. The present status of the inputs (actual values) is frozen when in this mode (Fig. 32).

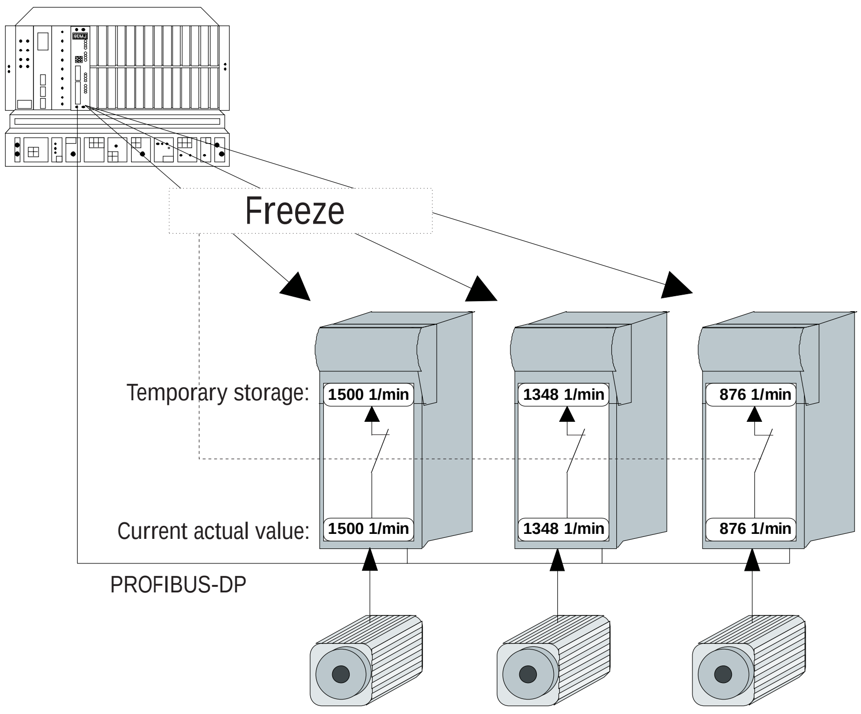


Fig. 32: Simultaneous freezing of the current actual values with the Freeze command

The master now has enough time to retrieve all the actual values from the servo controllers (Fig. 33). When a new Freeze command is received, all addressed servo controllers simultaneously save their current actual values (temporarily). The servo controllers quit Freeze mode when the *Unfreeze* control command is issued.

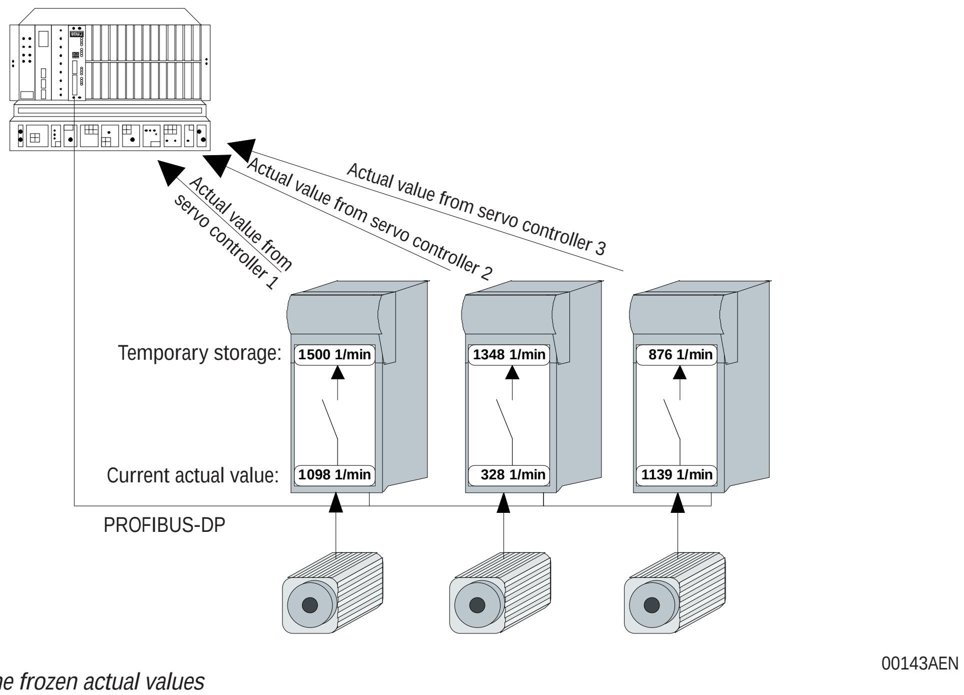


Fig. 33: Reading the frozen actual values

The MOVIDYN® Servo Controller supports both *Sync Mode* and *Freeze Mode*. This makes it possible to group together a number of servo controllers and synchronize them using the Sync and Freeze commands. The Sync command enables you to activate the setpoints on all drives simultaneously. Similarly, the Freeze command permits the actual values from all the drives on the bus system to be read in at the same time.

3.6 Control via PROFIBUS-DP

The servo controller is controlled via the process data channel, which can be one, two or three I/O words in length. These process data words are, for example when a programmable logic controller is being used as DP master, stored in the I/O or peripherals area of the control system and can thus be addressed in the usual manner (Fig. 34).

While the process input data (actual values) are being read, e.g. using the Load command in the case of Simatic S5, the process output data (setpoints) can be sent using the Transfer commands. Referring to Fig. 34, Example 1 shows the syntax for handling the process input and output data of the MOVIDYN® Servo Controller. The factory setting for the process data channel is shown as a comment.

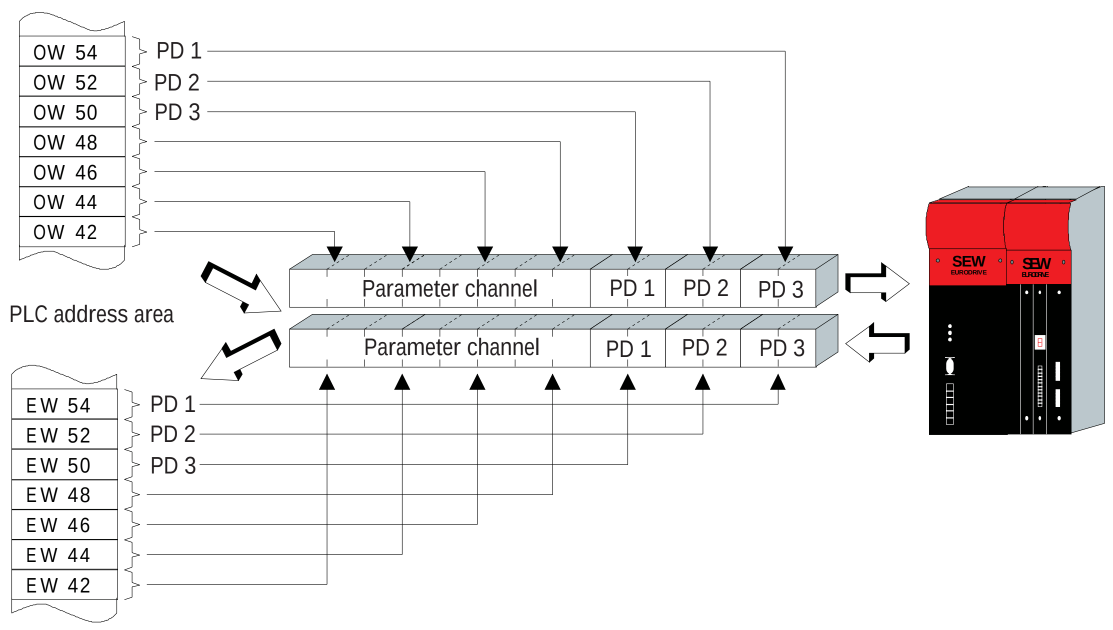
```
L IW 50         Load PD1 (status word 1)
L IW 52         Load PD2 (speed actual value)
L IW 54         Load PD3 (no function)

L KH 0006
T OW 50         Write 6hex to PD1 (control word 1 = enable)

L KF +1500
T OW 52         Write 1500 dec to PD2 (speed setpoint = 300 1/min)

L KH 0000
T OW 54         Write 0 hex to PD3 (no function, sent value without effect)
```

Example 1: Controlling the servo controller via the process data



00144AEN

Fig. 34: Allocation of the PLC I/O area

For details of the control via the process data channel, in particular the coding of the control and status words, please refer to the *Unit Profile User Manual.*

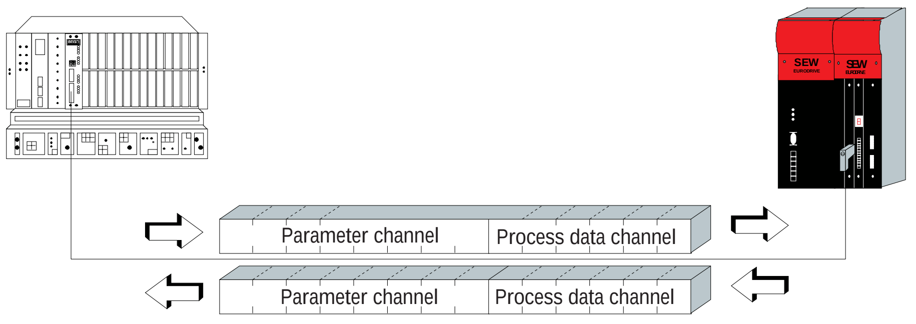
3.7 Parameterizing via PROFIBUS-DP

The servo controller parameters are read and written by the READ and WRITE services of the application layer (layer 7). If there is no layer 7, as in the case of PROFIBUS-DP, a suitable application layer must be emulated, i.e. mechanisms for parameterizing the servo controller be created.

3.7.1 Structure of the Parameter Channel

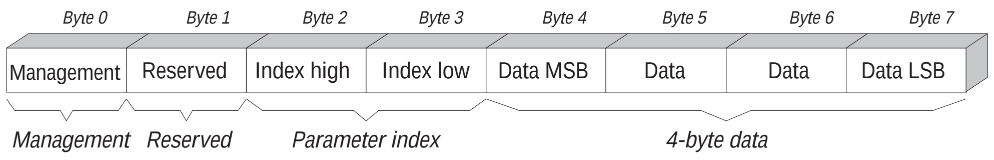
The parameterizing of field units using fieldbus systems, which do not provide an application layer, requires the emulation of the most important functions and services, such as READ and WRITE for the reading and writing of parameters. In the case of PROFIBUS-DP, this requires a Parameter Process Data Object (PPO) to be defined. This PPO is transferred cyclically and in addition to the process data channel contains a parameter channel through which acyclic parameter values can be transferred (Fig. 35).

Fig. 36 shows the structure of the parameter channel. It generally consists of a management byte, an index word, a reserved byte and four data bytes.



00145AEN

Fig. 35: Parameter process data object for PROFIBUS-DP

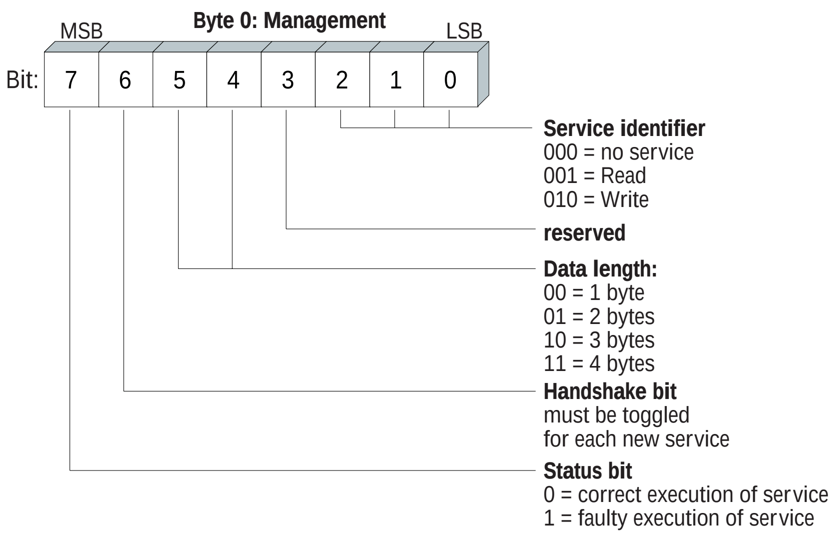


00146AEN

Fig. 36: Structure of the parameter channel

3.7.1.1 Management of the Parameter Channel

The entire parameter adjustment procedure is co-ordinated using byte 0: *Management*. This byte makes important parameters, such as service identifier, data length, version and status, of the executed service available. Fig. 37 shows that bits 0, 1 and 2 contain the service identifier, in other words, they define which service is to be executed. Bit 3 is currently reserved and should generally remain set to zero. Bit 4 and bit 5 contain the data length in bytes for the Write service, which in the case of SEW servo controllers should normally be set to 4 bytes.



00147AEN

Fig. 37: Structure of the management byte

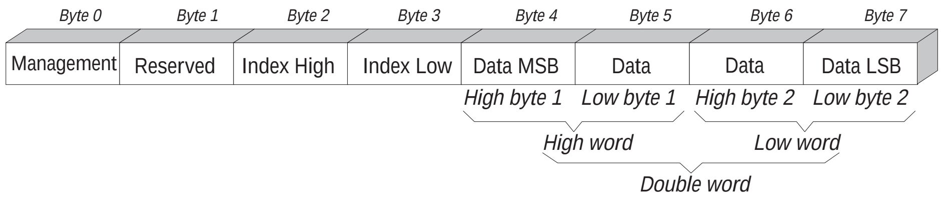
Bit 6 is used as a handshake between controller and servo controller. It initiates the execution of the transferred service in the servo controller. As the parameter channel is transferred in each cycle with the process data, particularly with PROFIBUS-DP, execution of the service in the servo controller must be edge controlled using the handshake bit 6. The value of this bit is therefore toggled each time a new service is to be executed. The servo controller uses the handshake bit to signal whether the service has been executed or not. The service is executed as soon as the controller notices that the received and transmitted handshake bits correspond. Status bit 7 indicates whether the service was executed properly or produced an error.

3.7.1.2 Index Addressing

Byte 2: *Index High* and byte 3: *Index Low* are used to identify the parameter to be read or written via the fieldbus system. The parameters of a servo controller are addressed using a standard index, irrespective of the type of fieldbus system. Byte 1 should be considered reserved and must generally be set to 0x00.

3.7.1.3 Data Area

As shown in Fig. 38 the data are contained in byte 4 to byte 7 of the parameter channel. This allows a maximum of 4-byte data to be transmitted per each service. The data are generally entered flush right, i.e. byte 7 contains the least significant data byte (data LSB), byte 4 correspondingly the most significant data byte (data MSB).

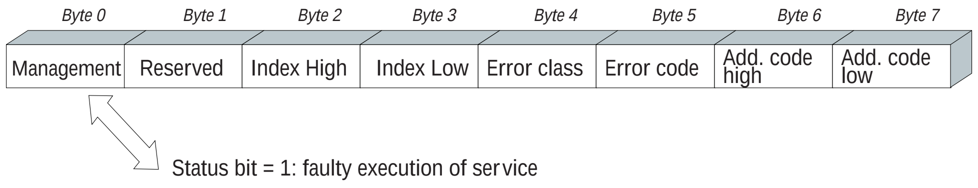


00148AEN

Fig. 38: Definition of the data area in the parameter channel

3.7.1.4 Faulty Execution of Service

Faulty execution of service is signalled by setting the status bit in the management byte. If the received handshake bit is identical to the transmitted handshake bit, the servo controller has executed the service. If the status bit indicates an error, the error code is entered in the data area of the parameter message (Fig. 39). Bytes 4-7 provide the Return Code in a structured format (see the section Return Codes).



00149AEN

Fig. 39: Structure of the parameter channel in the event of faulty execution of service

3.7.2 Reading a Parameter via PROFIBUS-DP (Read)

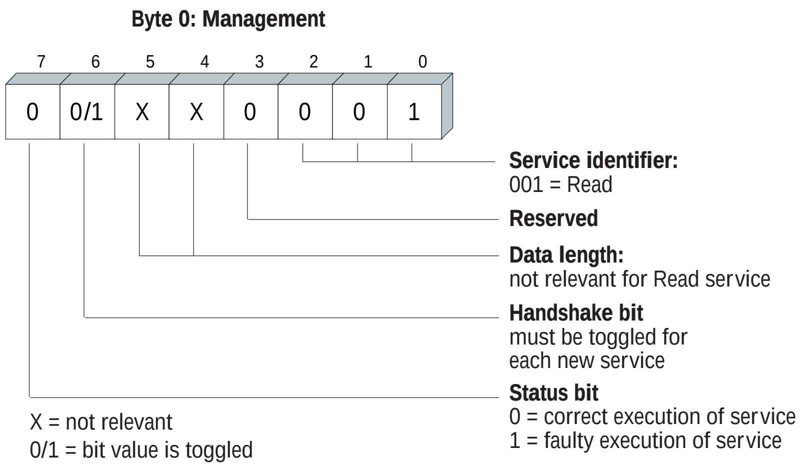
When executing a READ service via the parameter channel, the handshake bit should not be toggled until the entire parameter channel has been set up accordingly for the service in question, as the parameter channel is transferred on a cyclic basis. Adhere to the following sequence of operations to read a parameter:

1) Enter the index of the parameter to be read in byte 2 (Index High) and byte 3 (Index Low).

2) Enter the Service identifier for the Read service in the management byte (byte 0).

3) Transfer the Read service to the servo controller by toggling the handshake bit.

As this is a Read service, the transferred data bytes (bytes 4 ...7) and the data length (in the management byte) are ignored and therefore do not need to be entered. The servo controller now processes the Read service and returns the acknowledgement by toggling the handshake bit.


Byte 0: Management
7 6 5 4 3 2 1 0
0 0/1 X X 0 0 0 1
Service identifier: 001 = Read
Reserved
Data length: not relevant for Read service
Handshake bit must be toggled for each new service
Status bit 0 = correct execution of service 1 = faulty execution of service
X = not relevant
0/1 = bit value is toggled

00150AEN

Fig. 40: Coding of the READ service in the management byte

Fig. 40 shows how the READ service is coded in the management byte. The data length is not relevant so only the Service Identifier for the READ service has to be entered. The service is activated in the servo controller when the handshake bit is toggled. For example, the Read service could be activated by entering the codes 01_{hex} or 41_{hex} in the management byte.

3.7.3 Writing a Parameter via PROFIBUS-DP (Write)

When executing a WRITE service via the parameter channel, the handshake bit should not be toggled until the entire parameter channel has been set up accordingly for the service in question, as the parameter channel is transferred on a cyclic basis. Adhere to the following sequence of operations to write a parameter:

1) Enter the index of the parameter to be written in byte 2 (Index High) and byte 3 (Index Low).

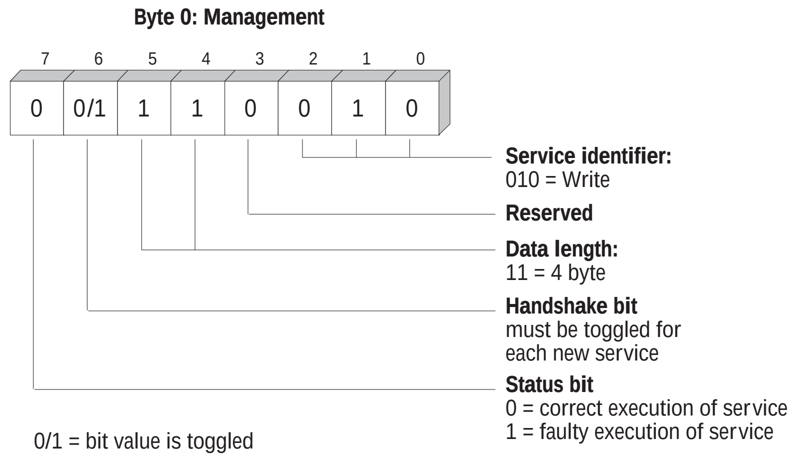
2) Enter the data to be written in bytes 4...7.

3) Enter the Service Identifier for the Write service in the management byte (byte 0).

4) Transfer the Write service to the servo controller by toggling the handshake bit.

The servo controller now processes the Write service and returns the acknowledgement by toggling the handshake bit.

Fig. 41 shows how the WRITE service is coded in the management byte. The data length for all SEW servo controllers is 4 bytes. Transfer of this service to the servo controller is by toggling the handshake bit. A WRITE service to SEW servo controllers therefore generally has the management byte code 32_{hex} or 72_{hex}.



00151AEN

Fig. 41: Coding of the WRITE service in the management byte

3.7.4 Sequence of Parameter Adjustment via PROFIBUS-DP

Using the WRITE service as an example, Fig. 42 shows the parameterizing sequence between control system and servo controller on PROFIBUS-DP. To simplify the sequence, only the management byte of the parameter channel is shown in Fig. 42.

While the controller sets up the parameter channel for the Write service, the servo controller simply receives and returns the parameter channel. The service is first activated when the handshake bit has changed, in this case from 0 to 1. The servo controller then looks at the parameter channel and processes the Write service, and responds to all messages, though with the handshake bit still = 0. Confirmation that the service has been executed is indicated by the change of the handshake bit in the response message from the servo controller. The control system recognizes that the received handshake bit is now the same as the one sent and can then prepare a new parameter adjustment.

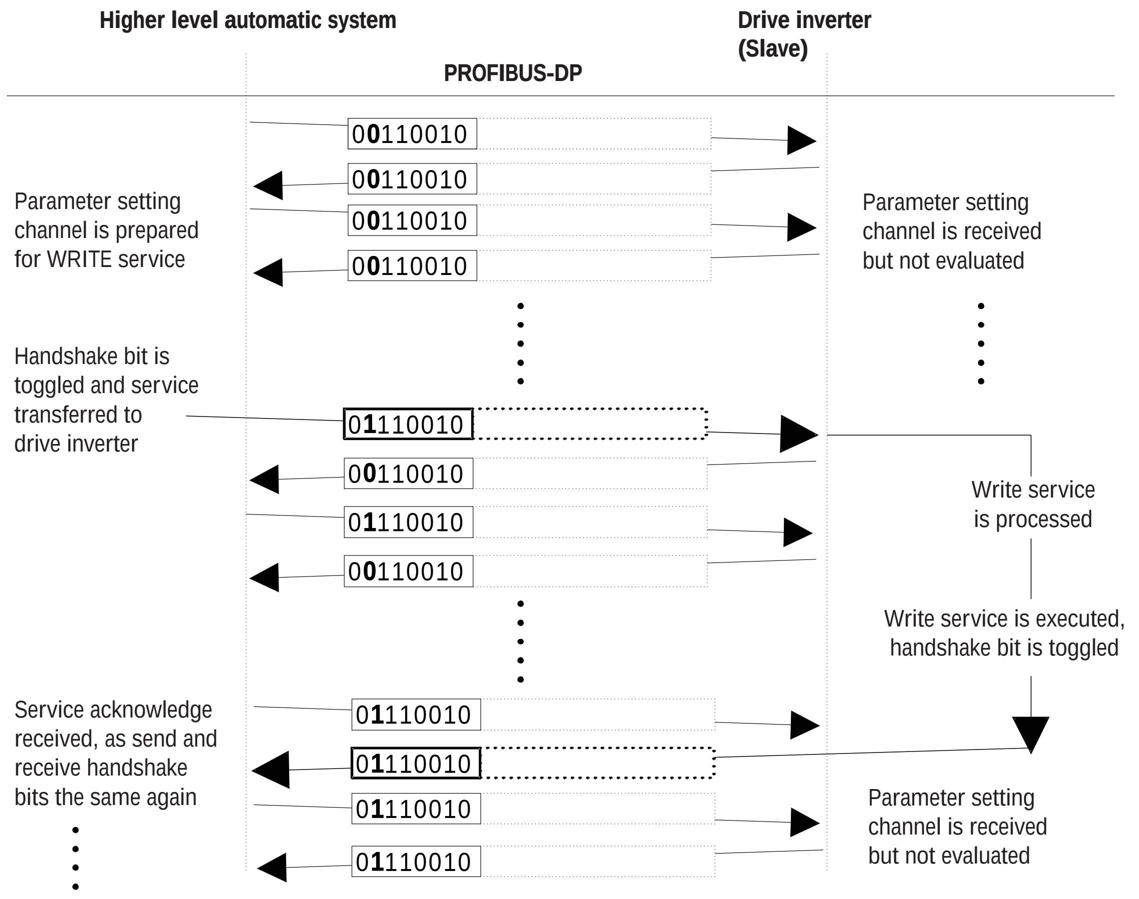

Higher level automatic system
Drive inverter (Slave)
PROFIBUS-DP
00110010
Parameter setting channel is prepared for WRITE service
Parameter setting channel is received but not evaluated
Handshake bit is toggled and service transferred to drive inverter
01110010
Write service is processed
Write service is executed, handshake bit is toggled
Service acknowledge received, as send and receive handshake bits the same again
Parameter setting channel is received but not evaluated
00152AEN

Fig. 42: Sequence of parameter adjustment via PROFIBUS-DP

3.7.5 Parameter Data Format

When parameterizing via the fieldbus interface the same parameter coding is used as when parameterizing via the serial interfaces RS-232 and RS-485. The majority of the parameters is transmitted in 4-byte BCD format. 32-bit values are directly entered in the parameter channel as 4-byte hex values.

For details of the data formats and value ranges of the individual parameters please refer to the SEW documentation MOVIDYN®*Parameter List*.

3.8 GSD Files

All slave-specific features are stored in a device database file (GSD file). EN 50170 V2 / DIN E 19245 Part 3 describes the format of a GSD file. It can be used by the DP master for easy configuration of the DP slave. However, as some DP masters do not support this file format, additional type files are required. These files are enclosed to the fieldbus documentation package on a diskette. In addition, these files can be downloaded via modem or the Internet at the addresses below:

Internet:
http://www.SEW-EURODRIVE.com (all files)
http://www.PROFIBUS.com (GSD files only)

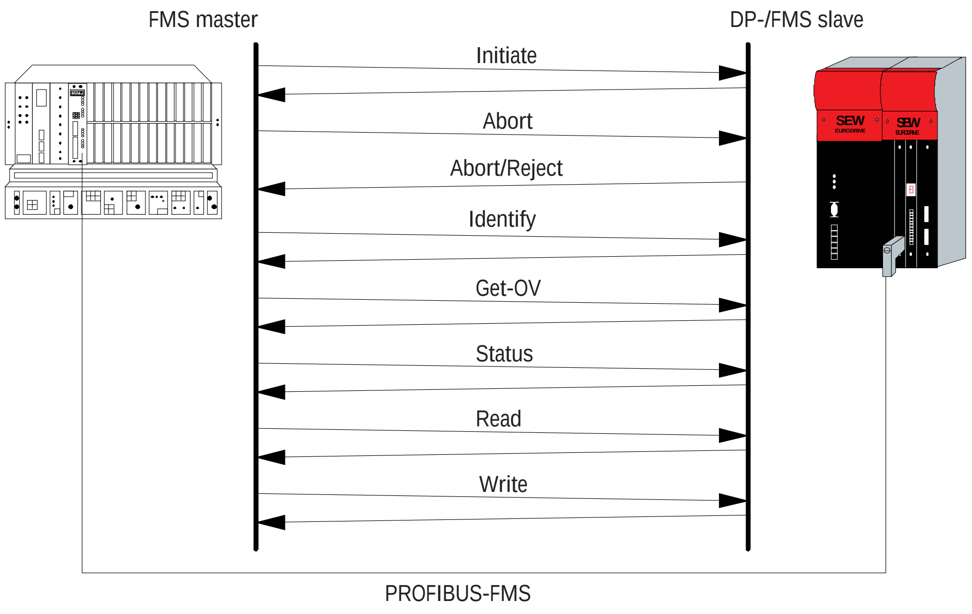
Modem:
Siemens Schnittstellencenter Fürth
Tel.: +49 (911) 737972 (GSD and Siemens type files)

4 The PROFIBUS-FMS Interface

With the AFP 11 option the MOVIDYN® Servo Controller offers a FMS interface conforming to DIN 19245 Part 2.

4.1 FMS Services

With the AFP 11 option, the MOVIDYN® Servo Controller supports the FMS services shown in Fig. 43. These FMS services conform to the definitions in the sensor/actuator profile.



00153AEN

Fig. 43: FMS services supported by the MOVIDYN®Servo Controller

4.1.1 Initiate

With the FMS service *Initiate* (establish link), a communications link is established between an FMS master and the MOVIDYN® Servo Controller.

The establishment of the link is always performed by the FMS master. As the link is being established, various conventions regarding the communications link are checked, e.g. FMS services supported, user data length, etc. If the link is successfully established, the servo controller answers with a positive *Initiate Response*.

If the link could not be established, then the conventions regarding the communications link between the FMS master and MOVIDYN® Servo Controller do not match. The servo controller will answer with an *Initiate Error Response*. In this event, compare the configured communications relationship list of the FMS master with that of the servo controller.

The attempt to establish an already existing communications link again generally leads to *Abort*. The communications link will then no longer exist so the FMS service *Initiate* will have to be performed again to reinstate the communications link.

4.1.2 Abort

An existing communications link between the FMS master and the MOVIDYN® Servo Controller is cleared using the FMS service *Abort*. *Abort* is an unacknowledged FMS service and can be initiated both by the FMS master as well as by the MOVIDYN®.

The attempt to establish an already existing communications link again generally leads to *Abort*. The communications link will then no longer exist so the FMS service *Initiate* will have to be performed again to reinstate the communications link.

4.1.3 Reject

With the FMS service *Reject*, the MOVIDYN® Servo Controller rejects an illegal FMS service. The servo controller indicates to the FMS master that this is an illegal or invalid service.

4.1.4 Identify

With the FMS service *Identify*, the MOVIDYN® Servo Controller passes the following data to the FMS master for definite identification:

```
vendor_name:    SEW-Eurodrive GmbH & Co
model_name:     MOVIDYN®
revision:       821XXXYYZZ    (Number of servo controller system software)
```

4.1.5 Get-OV

With the FMS service *Get-OV*, the FMS master can retrieve the object description of the MOVIDYN® Servo Controller. In general all drive parameters are described as communications objects. More precise information about object descriptions can be found in Section 4.2.

The MOVIDYN® Servo Controller supports both the short as well as the long form of the FMS service *Get-OV*.

4.1.6 Status

With the FMS service *Status,* the FMS master can check the logical communications status of the AFP 11 option of the MOVIDYN® Servo Controller. The *Local Detail* attribute is not supported by the servo controller.

4.1.7 Read

With the FMS service *Read,* the FMS master can read all the communications objects (drive parameters) of the MOVIDYN® Servo Controller. All drive parameters as well as their codes are listed in detail in the documentation *MOVIDYN®Parameter List.*

4.1.8 Write

With the FMS service *Write*, the FMS master can write all the drive parameters of the MOVIDYN®. If a drive parameter is assigned an invalid value (e.g. value too high), the servo controller generates a *Write Error Response* giving the precise cause of the error (see Section Return Codes).

4.2 Object List

With the FMS services *Read* and *Write*, the FMS master can access all the communications objects defined in the object list.

All drive parameters that can be accessed via the bus system are described as communications objects in the static object list. All objects in the static object list are addressed via a fieldbus index. Table 3 shows the structure of the object list of the MOVIDYN® Servo Controller.

Normally, the whole object list is always generated when the servo controller is switched on. To also be able to guarantee full access to all parameters via PROFIBUS-FMS if additional drive parameters are added in the future, the generated object list is larger than the number of drive parameters implemented. Access to objects that cannot be directly mapped to a drive parameter is rejected with a negative response. The index area is divided into two logical areas. The drive parameters are addressed with indices from $1000_{dec.}$ The parameter index can be obtained from the SEW manual *MOVIDYN® Parameter List.* Indices below 1000_{dec} are handled directly by the PROFIBUS option pcb.

Fieldbus index (decimal)	Name of the communications object
988	1 process output data word (1 PO)
989	2 process output data words (2 PO)
990	3 process output data words (3 PO)
991	1 process input data word (1 PI)
992	2 process input data words (2 PI)
993	3 process input data words (3 PI)
994	Min Tsdr
995	DP station diagnosis data (SlaveDiag)
996	Download Parameter Block
997	Universal Write parameter
998	Universal Read pointer
999	Universal Read parameter
1000 + Parameter index	Drive parameter for MOVIDYN® (Parameter index see SEW documentation MOVIDYN® *Parameter List*)

Table 3: Structure of the MOVIDYN® static object list

4.2.1 Object Description of the Drive Parameters

The drive parameters of the MOVIDYN® Servo Controller are described in detail in the SEW documentation *MOVIDYN® Parameter List*. In addition to the parameter index, i.e. the number with which you can address the appropriate parameter via the communications interfaces of the servo controller, you will find further information about the coding, range of values and meaning of the parameter data. To access all drive parameters via PROFIBUS-FMS, you must add the value 1000dec to the index shown in the parameter list to access the fieldbus index. In general, you can read or write drive parameters with the following formula:

Fieldbus Index = Parameter Index + 1000_{dec}

The object description in the object list is identical for all drive parameters. Even parameters that can only be read are given the attribute Read All/Write All in the object list, as the servo controller itself carries out the appropriate testing and if necessary supplies a return code. Table 4 shows the object descriptions of all drive parameters.

Index:	Parameter Index + 1000_{dec}
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	4
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 4: Object description of the MOVIDYN® drive parameters

4.2.2 Objects for Process Data Communication

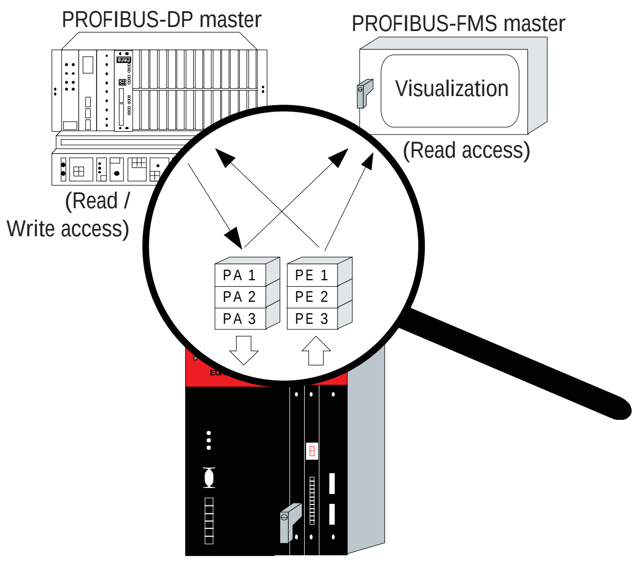
For process data communication via FMS six communications objects are available which are described in Table 5.

Fieldbus index	Designation	Functionality
988	1 process output data word (1 PO)	1) Only FMS mode: Transmission of a process output data word from the master to the servo controller with the FMS service *Write*. 2) In the mixed mode (DP/FMS): Reading a process output data word specified by the DP master (e.g. for visualization) with the FMS service *Read.*
989	2 process output data words (2 PO)	1) Only FMS mode: Transfer of two process output data words from the master to the servo controller with the FMS service *Write*. 2) In mixed mode (DP/FMS): Reading two process output data words specified by the DP master (e.g. for visualization) with the FMS service *Read.*
990	3 process output data words (3 PO)	1) Only FMS mode: Transfer of three process output data words from the master to the servo controller with the FMS service *Write.* 2) In mixed mode (DP/FMS): Reading three process output data words specified by the DP master (e.g. for visualization) with the FMS service *Read.*
991	1 process input data word (1 PI)	Reading one process input data word (PD 1) with the FMS service *Read.*
992	2 process input data words (2 PI)	Reading two process input data words (PD1, PD 2) with FMS service *Read.*
993	3 process input data words (3 PI)	Reading three process input data words (PD1, PD2, PD3) with FMS service *Read*.

Table 5: Functionality of the process data objects

In straight PROFIBUS-FMS mode, an FMS master can control the MOVIDYN® Servo Controller via the process data channel using the communications objects listed in Table 5. The process output data are transferred during this process to the appropriate process output data object by the Write service. Process input data are read by the Read service into the relevant process input data object. Whereas the process input data objects can generally only be read, the process output data objects have both Read and Write access. So in mixed mode (DP/FMS), for example, the process output data sent by the DP master can be read and visualized by the FMS master.

The data consistency required for the exchange of data via PROFIBUS-FMS is achieved by providing the appropriate communications objects for each process data length. With a process data length of 3, for example, process data exchange will only be consistent with the objects “3 PI” and “3 PO”. Fig. 44 shows the various ways of accessing communications objects in mixed mode (DP/FMS).



00154AEN

Fig. 44: Process data access of DP master and FMS master in mixed mode (DP/FMS)

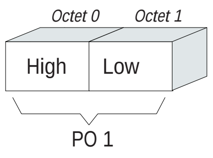
4.2.2.1 Process Output Data Objects

Tables 6 to 8 show the communications objects for the process output data (setpoints from master to servo controller). In straight FMS mode, the FMS master can use the FMS service *Write* to write these objects and thus control the servo controller via the process data channel. Furthermore, in mixed mode (DP/FMS) an FMS master can use the FMS service *Read* to read (and if necessary visualize) the setpoints specified by a DP master via PROFIBUS-DP.

Index:	988
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	2 bytes
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 6: Description of the object “1 process output data word (1 PO)”

Fig. 45 shows the structure of the object “1 process output data word (1 PO)”.



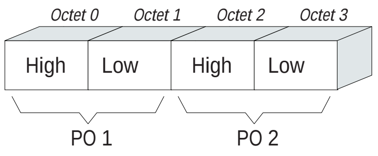
00155AEN

Fig. 45: Structure of the object “1 process output data word (1 PO)”

Index:	989
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	4 bytes
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 7: Description of the object "2 process output data words (2 PO)"

Fig. 46 shows the structure of the object "2 process output data words (2 PO)".



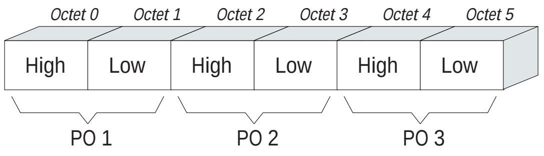
00156AEN

Fig. 46: Structure of the object "2 process output data words (2 PO)"

Index:	990
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	6 bytes
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 8: Description of the object "3 process output data words (3 PO)"

Fig. 47 shows the structure of the object "3 process output data words (3 PO)".



00157AEN

Fig. 47: Structure of the object "3 process output data words (3 PO)"

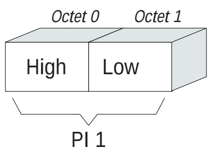
4.2.2.2 Process Input Data Objects

Tables 9 to 11 show the communications objects for the process input data (actual values of the servo controller). These objects can only be read with the FMS service *Read*.

Index:	991
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	2 bytes
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 9: Description of the object "1 process input data word (1 PI)"

Fig. 48 shows the structure of the object "1 process input data word (1 PI)".



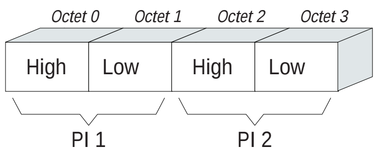
00158AEN

Fig. 48: Structure of the object "1 process input data word (1 PI)"

Index:	992
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	4 bytes
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 10: Description of the object "2 process input data words (2 PI)"

Fig. 49 shows the structure of the object "2 process input data words (2 PI)".



00159AEN

Fig. 49: Structure of the object "2 process input data words (2 PI)"

Index:	993
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	6 bytes
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 11: Description of the object “3 process input data words (3 PI)”

Fig. 50 shows the structure of the object “3 process input data words (3 PI)”.

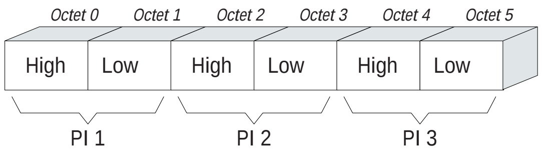


Fig. 50: Structure of the “3 process input data words (3PI)” object

4.2.3 “Min Tsdr” Object

Where several PROFIBUS masters are present, it is often necessary to modify the response delay time (min T_{SDR}). This has to be done when the servo controller responds faster than the master can switch between send and receive. DIN 19245 defines default values with which every PROFIBUS master or slave in a PROFIBUS network can safely be operated.

This minimum response delay time for PROFIBUS is set using the DIP switch on the option pcb (see Fig. 8). This DIP switch is used to toggle between the *min T_{SDR}* default value for straight DP applications and the *min T_{SDR}* default value for mixed FMS/DP applications. The default values for the minimum response delay time defined in DIN 19245 will be chosen depending on the selected baud rate.

The FMS object “Min Tsdr” can then be used to read or write the *min T_{SDR}* bus parameter directly. When changing *min T_{SDR}* remember that when the servo controller is powered up again (mains supply and 24V supply ON/OFF), however, the *min T_{SDR}* default value will again be operative.

Index:	994
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	1 byte
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 12: Description of the “Min Tsdr” object

Note:

Changing *min T_{SDR}* can cause major malfunctions across the entire PROFIBUS network and should therefore only be done by PROFIBUS experts. As a rule, the default setting according to DIN 19245, which is set using the DIP switch on the option pcb, is more than adequate. These DIN 19245 default values guarantee stable operation of the PROFIBUS network.

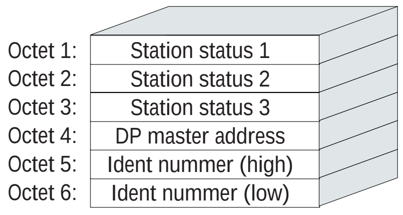
4.2.4 “DP Station Diagnosis” Object

The diagnostic messages of the servo controller in DP mode are stored in this object. The DP master can retrieve these diagnostic data using the DP service DDLM_SlaveDiag. An FMS master can retrieve these diagnostic messages via this communications object using the FMS service *Read*. Table 13 provides a definition of this communications object.

Index:	995
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	6 bytes
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 13: Description of the “DP Station Diagnosis” object

The “DP Station Diagnosis” object consists of six octets, structured as shown in Fig. 52. The contents of the individual octets conform to DIN E 19245 (Part 3) and are not discussed here.

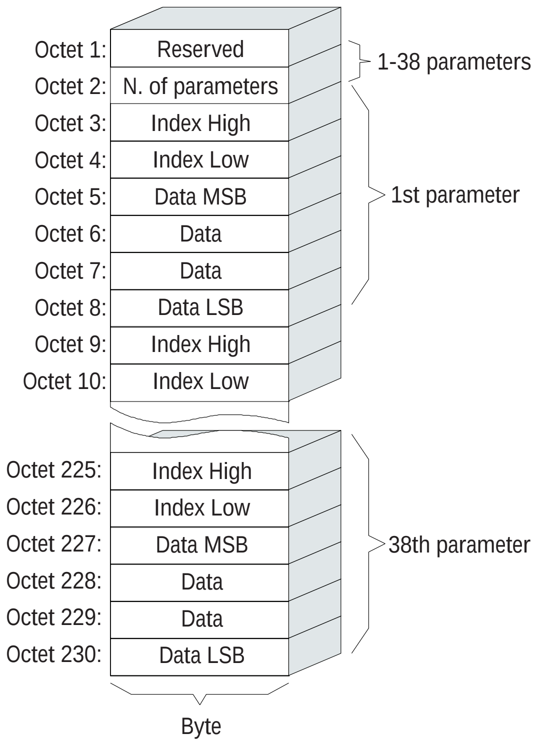


00161AEN

Fig. 51: Structure of the “DP Station Diagnosis” object

4.2.5 “Download Parameter Block” Object

The “Download Parameter Block” object enables a maximum of 38 MOVIDYN® drive parameters to be written at the same time. This means you can use this object to parameterize the servo controller in the start-up phase with only one *Write* service call. Since, as a rule, only a few parameters have to be altered, this parameter block with a maximum of 38 parameters is adequate for almost all applications. The user data area is fixed at 38 x 6 + 2 bytes = 230 bytes (octet string type). Fig. 52 shows the structure of the “Download Parameter Block” object.



00162AEN

Fig. 52: Structure of the "Download Parameter Block" object

The "Download Parameter Block" object is only handled locally on the fieldbus option pcb and is defined as shown in Table 14.

Index:	996
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	230
Local address:	-
Password:	-
Access groups:	-
Access rights:	Write all
Name[16]:	-
Extension length:	-

Table 14: Description of the "Download Parameter Block" object

With the WRITE service to the "Download Parameter Block" object, a parameterization mechanism is started in the fieldbus option pcb that successively transmits to the servo controller all the parameters in the user data area of the object.

After successfully processing the Download Parameter Block, i.e. all parameters transferred from the FMS-master have been written, the Write service is ended with a positive Write Response.

In the event of an error, a negative Write Response is returned. In this event, the return code will contain more precise details about the type of error and, in addition, the parameter number (1..38) where the error occurred (see Example 2).

Example 2: Error writing the 11th parameter

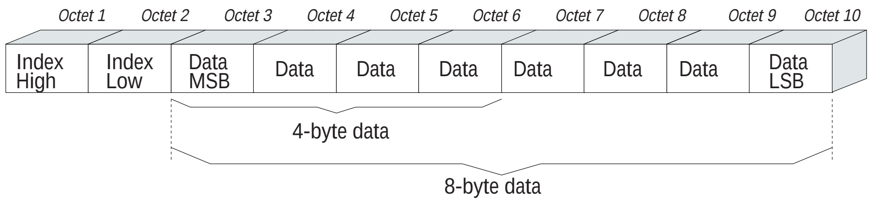
Write Error Response:		
Error Class:	8	Other
Error Code:	0	Other
Additional Code High:	11dec	Error writing parameter 11
Additional Code Low:	15 hex	Value too large

If an error occurs when a parameter is written, processing of the parameter block is aborted. All parameters in the block following the faulty parameter are not transmitted to the servo controller and remain unchanged.

4.2.6 "Universal Write Parameter" Object

This object permits any parameter to be written, regardless of the size and content of the object list on the fieldbus option pcb.

The parameter value to be written is shown together with the index in a 10-byte data area of the "Universal Write" object. The parameter values can be four or eight bytes long depending on the drive parameter. The length can be obtained from the current parameter list for the respective unit. The parameter data must be entered flush left in every case (Fig. 53).



00163AEN

Fig. 53: Structure of the "Universal Write" object

The "Universal Write" object is only handled locally on the fieldbus option pcb and is defined as shown in Table 15.

Index:	997
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	10
Local address:	-
Password:	-
Access groups:	-
Access rights:	Write all
Name[16]:	-
Extension length:	-

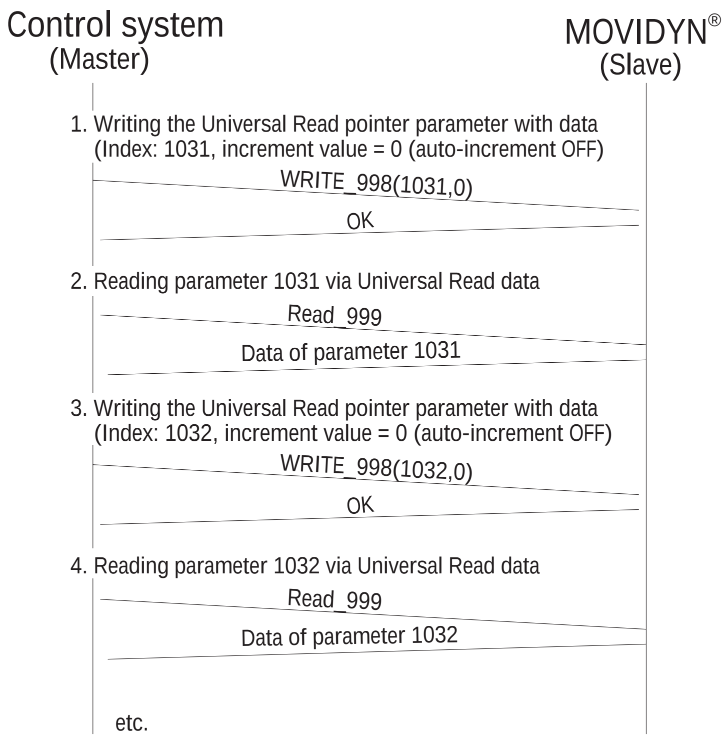
Table 15: Description of the "Universal Write" object

4.2.7 "Universal Read" Functionality Objects

The Universal Read objects form the counterpart to Universal Write. The Universal Read objects permit reading of any parameter independent of the object list being used. The execution of a Universal Read takes place using both the "Universal Read Pointer" and "Universal Read Data" objects.

The fieldbus index (read pointer) to be read by the servo controller is first entered in the "Universal Read Pointer" object using the FMS service *Write*. The value of the drive parameter is then read via the "Universal Read Data" object with the FMS service *Read*. To avoid having to rewrite the read pointer after each operation when reading a consecutive series of parameters, the Universal Read also has an auto-increment function where the read pointer ("Universal Read Pointer" object) is incremented by a specified amount each time the "Universal Read Data" object is read. This number is set together with the read pointer and stored in the "Universal Read Pointer" object.

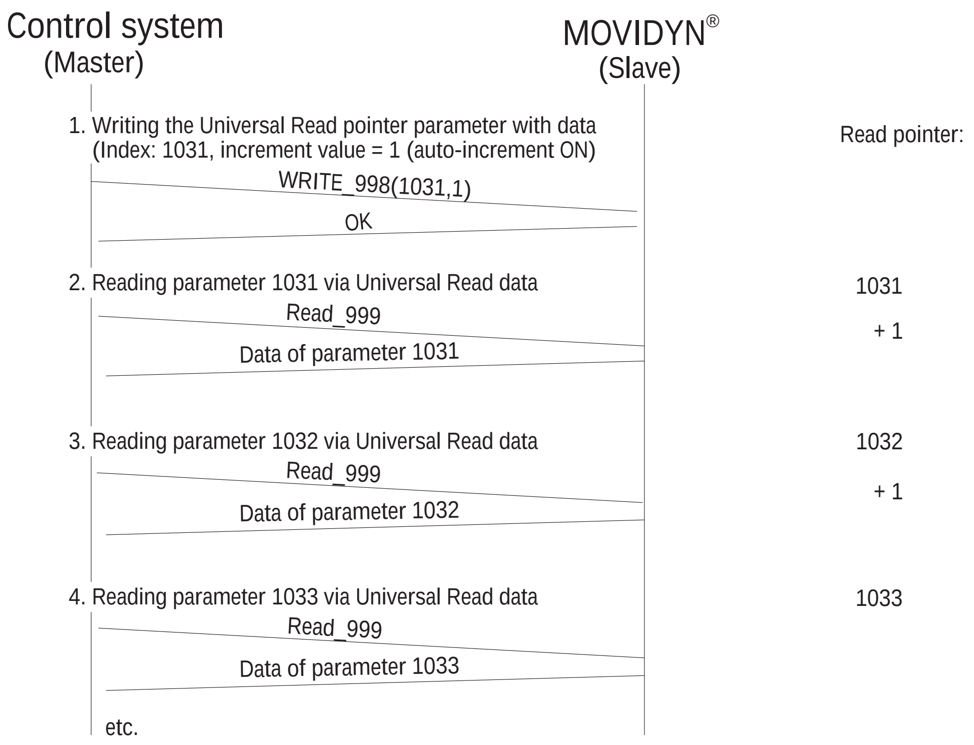
Fig. 54 shows an example of how Universal Read works without the auto-increment function.


Control system
(Master)
MOVIDYN®
(Slave)
1. Writing the Universal Read pointer parameter with data
(Index: 1031, increment value = 0 (auto-increment OFF)
WRITE_998(1031,0)
OK
2. Reading parameter 1031 via Universal Read data
Read_999
Data of parameter 1031
3. Writing the Universal Read pointer parameter with data
(Index: 1032, increment value = 0 (auto-increment OFF)
WRITE_998(1032,0)
OK
4. Reading parameter 1032 via Universal Read data
Read_999
Data of parameter 1032
etc.

00164AEN

Fig. 54: Universal Read service without auto-increment function

Fig. 55 shows an example of how Universal Read works using the auto-increment function.

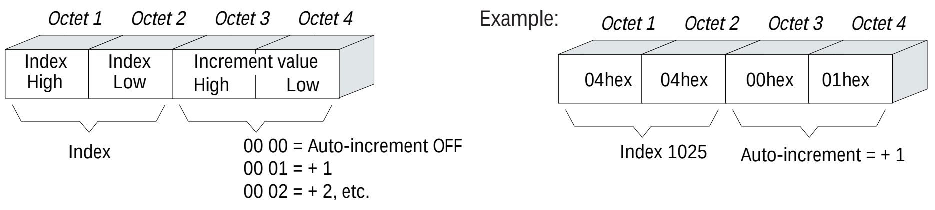

Control system
(Master)
MOVIDYN®
(Slave)
1. Writing the Universal Read pointer parameter with data (Index: 1031, increment value = 1 (auto-increment ON)
WRITE_998(1031,1)
OK
Read pointer:
2. Reading parameter 1031 via Universal Read data
Read_999
Data of parameter 1031
1031
+ 1
3. Reading parameter 1032 via Universal Read data
Read_999
Data of parameter 1032
1032
+ 1
4. Reading parameter 1033 via Universal Read data
Read_999
Data of parameter 1033
1033
etc.

00165AEN

Fig. 55: Universal Read service with auto-increment function

4.2.7.1 “Universal Read Pointer” Object

The “Universal Read Pointer” object contains within its 4 data bytes both the fieldbus index to be read as a read pointer as well as the number used in auto-increment mode. Fig. 56 shows the structure of this object.



00166AEN

Fig. 56: Structure of the Universal Read Pointer parameter

When the auto-increment mode is active (increment value greater than 0), the index is increased after reading the “Universal Read Data” object by the predefined increment value. The default value of this object is

Index:	**1000dec**
Auto increment:	0 = OFF

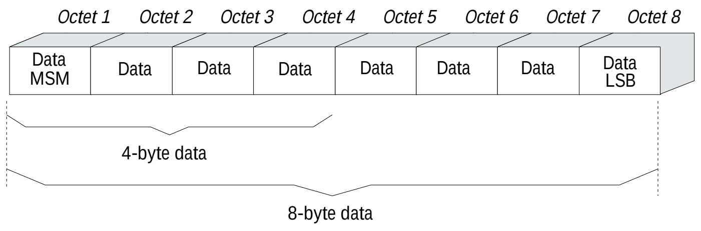
The auto-increment value is generally treated as having no sign, i.e. the value is generally added. The “Universal Read Pointer” object is only handled locally on the fieldbus option pcb and is defined as shown in Table 16.

Index:	998
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	4
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 16: Description of the “Universal Read Pointer” object

4.2.7.2 “Universal Read Data” Object

Accessing this parameter using the FMS service *Read* returns the parameter value of the read pointer held in the “Universal Read Pointer” object. Fig. 57 shows the structure of this object.



00167ADE

Fig. 57: Structure of the Universal Read Data parameter

The number of valid data can be determined from the Parameter List. Data are generally entered flush left, i.e. beginning with the most significant byte in octet 1.

The “Universal Read Data” object is only handled locally on the fieldbus option pcb and is defined as shown in Table 17.

Index:	999
Object code:	7 (Simple variable)
Data type index:	10 (Octet string)
Length:	8
Local address:	-
Password:	-
Access groups:	-
Access rights:	Read all / Write all
Name[16]:	-
Extension length:	-

Table 17: Description of the Universal Read Data object

4.3 Communications Relationship List (CRL)

The communications relationships between the MOVIDYN® Servo Controller and the FMS master are stored in the Communications Relationship List (CRL). You will need these CRL data to configure an FMS master that is to communicate with the MOVIDYN® Servo Controller via PROFIBUS-FMS.

4.3.1 CRL Definition

The communications relationship lists for the PROFIBUS-FMS contain various elements of definition which are briefly discussed below. For a more detailed explanation please refer to DIN 19245 Part 2.

4.3.1.1 Communications Reference (CREF)

All the links contained in the Communications Relationship List are numbered sequentially with a Communications Reference CREF. In accordance with the sensor/actuator profile, the CRL for the MOVIDYN® Servo Controller contains the Communications References CREF2 to CREF10. These CREFs represent the logical communications links from the point of view of the MOVIDYN® Servo Controller.

The CREFs in the CRL of the servo controller have no significance as far as configuring on the FMS master is concerned, as they describe the communications relationships from the point of view of the servo controller. The FMS master defines its communications relationships in its own CRL.

4.3.1.2 Link Type (TYPE)

The *TYPE* field in the CRL defines the type of link between two PROFIBUS stations. A distinction is generally made between a master-master and a master-slave communications link. As the MOVIDYN® Servo Controller is to be considered a PROFIBUS slave station, the CRL only contains the link types for the master-slave relationship. Table 18 shows the link types supported by the MOVIDYN® Servo Controller.

TYPE	Meaning
MSAZ	Master-slave link for acyclic data transfer without slave initiative
MSAZ_SI	Master-slave link for acyclic data transfer with slave initiative
MSZY	Master-slave link for cyclic data transfer without slave initiative
MSZY_	Master-slave link for cyclic data transfer with slave initiative

Table 18: Link types between FMS master and MOVIDYN®

4.3.1.3 Link Attribute (ATTR)

The *ATTR* link attribute indicates whether the link is an **o**pen (o) or a **d**efined communications link. In the case of an open link, the address (RADR) and the service access point (RSAP) of the communication partner are only entered as the link is being established. To ensure the servo controller will work properly with the various FMS masters (PROFIBUS conformance), all communication links of the MOVIDYN® Servo Controller are implemented as open links.

4.3.1.4 Service Access Points (LSAP, RSAP)

Service access points form the interface between the application layer (layer 7) and the data link layer (layer 2) of a PROFIBUS station across which messages are transferred.

From the point of view of the servo controller, the *LSAP* (*Local Link Service Access Point*) is the local service access point of the MOVIDYN® Servo Controller where the message crosses the interface between layer 2 and layer 7. Consequently, the *RSAP* (*Remote Service Access Point*) as seen by the MOVIDYN® Servo Controller is the service access point of the FMS master where the message crosses the interface between layer 2 and layer 7 of the FMS master.

As the servo controller does not know the *RSAP*, it is entered automatically only as the link is being established. The CRL for the servo controller therefore contains the entry *All*.

4.3.1.5 Station Address of the FMS Master (RADR)

The station address of the FMS master wishing to communicate with the servo controller via this communications relationship is entered in the RADR (Remote Address) field. As the address of the FMS master can change, it is only entered as the link is established. The CRL for the servo controller therefore contains the entry *All.*

4.3.1.6 Flow Control Counters (SCC, RCC, SAC, RAC)

The flow control counters indicate the maximum number of services running in parallel. Table 19 shows the meaning of the individual CRL entries.

Abbreviation	Meaning	
SCC	Send Confirmed Request Counter	Number of parallel confirmed services send
RCC	Receive Confirmed Request Counter	Number of parallel confirmed services receive
SAC	Send Acknowledged Request Counter	Number of parallel unconfirmed services send
RAC	Receive Acknowledge Request Counter	Number of parallel unconfirmed services receive

Table 19: Flow control counters

4.3.1.7 Control Interval Times (ACI, CCI)

These CRL entries specify the time intervals used in the monitoring of a communications link, i.e. the servo controller checks whether any data was transferred during the specified time interval. If this is not the case, communication is aborted. The time interval applies to both cyclic (Cyclic Control Interval, CCI) as well as acyclic links (Acyclic Control Interval, ACI).

4.3.1.8 Protocol Data Unit Size (max PDU Size)

This CRL entry indicates the maximum size of the protocol data unit (max PDU size). It comprises four entries as per table 20.

Abbreviation	Meaning
Send HiPrio	Maximum size of the protocol data unit for high priority send messages
Send LoPrio	Maximum size of the protocol data unit for low priority send messages
Rec. HiPrio	Maximum size of the protocol data unit for high priority receive messages
Rec. LoPrio	Maximum size of the protocol data unit for low priority receive messages

Table 20: Maximum PDU size

4.3.1.9 Supported FMS Services (Features Supported)

This CRL entry specifies which services the MOVIDYN® Servo Controller supports in the relevant communications relationship.

4.3.2 Communications Relationship Lists of the Servo Controller

The following tables show the individual communications relationships supported by the MOVIDYN® Servo Controller. Although the servo controller does not use any Event services, i.e. the servo controller cannot execute any slave initiatives, communication links with slave initiatives are supported. Even though the *Physical Read* and *Physical Write* services are supported according to the CRL, no *Physical Write* access can be performed.

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
2	MSZY	0	20	All	All	0	0	0	0	3000

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	0	00 00 00 00 20 00	Read.indication
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 21: CRL for master-slave, cyclic, Read

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
3	MSZY	0	21	All	All	0	0	0	0	3000

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	0	00 00 00 00 10 00	Write.indication
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 22: CRL for master-slave, cyclic, Write

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
4	MSZY_SI	0	22	All	All	0	0	1	0	3000

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	241	00 00 10 00 20 00	Read.indication Event-Notification.request*
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 23: CRL for master-slave, cyclic, with slave initiative, Read

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
5	MSZY_SI	0	23	All	All	0	0	1	0	3000

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	241	00 00 10 00 10 00	Write.indication Event-Notification.request*
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 24: CRL for master-slave, cyclic, Read

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
6	MSZY_SI	0	24	All	All	0	1	1	0	0

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	0	00 00 10 80 33 06	Read.ind Write.ind Phys.-Read.ind* Phys.-Write.ind* Get-OV-long.indication Event-Notification.request* Acknowledge-Event-Notification.ind* Alter-Event-Condition-Monitoring.ind*
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 25: CRL for master-slave, acyclic, with slave initiative

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
7	MSAZ	0	25	All	All	0	1	0	0	0

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	0	00 00 00 80 33 00	Read.ind Write.ind Phys.-Read.ind* Phys.-Write.ind* Get-OV-long.indication
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 26: CRL for master-slave, acyclic

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
8	MSAZ	0	26	All	All	0	1	0	0	0

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	0	00 00 00 80 33 06	Read.ind Write.ind Phys.-Read.ind* Phys.-Write.ind* Get-OV-long.indication Acknowledge-Event-Notification.ind* Alter-Event-Condition-Monitoring.ind*
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 27: CRL for master-slave, acyclic, with event notification for cyclic connections

Though FMS services for event processing which are marked with an (*) are offered in the CRL, they are not supported by the MOVIDYN® Servo Controller.

4.4 Communications Relationship List of the Master

A CRL corresponding to the CRL in the servo controller must be configured in the FMS master to be able to communicate with the MOVIDYN® Servo Controller via PROFIBUS-FMS. The master CRL must conform to the following conventions:

1) The FMS master may expect no more services from the slave than supported by the latter. The “Features supported” parameter may therefore only contain the service primitives (Request) that are defined as indications in the corresponding slave CRL.

2) The size of the Receive PDU (Rec. HiPrio, Rec.LoPrio) of the master must at least be that of the Send PDU (Send HiPrio, Send LoPrio) of the slave.

3) The corresponding flow control counters must agree (SCC_{Master} = RCC_{Slave} and RCC_{Master} = SCC_{Slave}).

Table 28 shows a communications relationship list specified in the master referring to a servo controller with the station address 8 and the CREF 7.

CREF	Type	ATTR	LSAP	RSAP	RADR	SCC	RCC	SAC	RAC	ACI/CCI
3	MSAZ	D	NIL	25	8	1	0	0	0	0

max PDU Size:		Features supported	Supported FMS services
Send HiPrio	0	00 00 00 80 33 06	Read.req Write.req Phys.-Read.req* Phys.-Write.req* Get-OV-long.req
Send LoPrio	241		
Rec. HiPrio	0		
Rec. LoPrio	241		

Table 28: Example of a master CRL for an acyclic master-slave link

5 Parameter Adjustment Return Codes

If parameters are adjusted incorrectly, different return codes are sent back from the servo controller to the parameterizing master, providing detailed information about the cause of the error. These return codes are generally structured according to DIN 19245 Part 2. A distinction is made between the following elements

error class
error code
additional code.

These return codes are described in detail in the *Fieldbus Profile User Manual* and are not part of this documentation. However, the following special cases can arise in connection with a PROFIBUS-FMS/DP:

5.1 Incorrect Service Code in the Parameter Channel

An incorrect service was specified when parameterizing the servo controller via the parameter channel. Only the READ and WRITE services are supported. Table 29 shows the return code for this special case.

	Code (dec)	**Meaning**
Error class:	5	Service
Error code:	5	Illegal parameter
Add. code high:	0	-
Add. code low:	0	-

Table 29: Return code in the case of incorrect service coding via the parameter channel

Error rectification:

Check bits 0..2 in the management byte of the parameter channel. Only the entries 001_{bin} for the READ service and 010_{bin} for the WRITE service are permitted.

5.2 Incorrect Specification of the Data Length in the Parameter Channel

When parameterizing via the parameter channel a data length not equal 4 data bytes was specified in the READ or WRITE service. Table 30 shows the return code.

	Code (dec)	**Meaning**
Error class:	6	Access
Error code:	8	Type conflict
Add. code high:	0	-
Add. code low:	0	-

Table 30: Return code for incorrect length in the parameter channel (length $\neq$ 4)

Error rectification:

Check bit 4 and bit 5 for the data length in the management byte of the parameter channel. Both bits must be 1.

5.3. Internal Communications Error

The return code shown in Table 31 is returned if a communications error has occurred between the option pcb and the servo controller system. It may be that the parameter adjustment service transferred via fieldbus was not performed and should be repeated. If this error recurs the servo controller must be switched off and then on again to reinitialize the unit.

	Code (dec)	**Meaning**
Error class:	6	Access
Error code:	2	Hardware Fault
Add. code high:	0	-
Add. code low:	0	-

Table 31: Return code if an internal communications error has occurred

<u>Error rectification:</u>

Repeat the READ or WRITE service. If the error recurs you should briefly disconnect the servo controller from the mains supply and then switch it on again. If the error persists, consult the SEW Service Department.

6 Technical Data

Technical data for the *AFP 11* option

For use with MOVIDYN® 51.. and higher

Profibus protocol options:
- PROFIBUS-FMS to DIN 19245 Part 2
- PROFIBUS-DP to DIN E 19245 Part 3
- Mixed mode PROFIBUS-FMS/DP (Combislave)

Auto-baud detect for:
- 9.6 kBaud
- 19.2 kBaud
- 93.75 kBaud
- 187.5 kBaud
- 500 kBaud
- 1500 kBaud

Connection technology:
- 9-pin type D connectors
- Pin assignment to DIN 19245 Part 1

Bus termination:
- Connectable for cable type A (up to 1500kBaud) to DIN E 19245 Part 3

Station address:
- 0-125 settable via DIP switch

Default bus parameter:
- Min T_{SDR} for FMS/DP and DP mode selectable via DIP switch

Name of DDB file:
- SEW_5100.GSD

DP Ident Number:
- 5100hex = 20736dec

DP configurations for DDLM_Chk_Cfg:
- $F0_{hex}$ = 1 process data word (1 I/O word)
- $F1_{hex}$ = 2 process data words (2 I/O words)
- $F2_{hex}$ = 3 process data words (3 I/O words)
- $F3_{hex}$, $F0_{hex}$ = parameter channel + 1 process data word (5 I/O words)
- $F3_{hex}$, $F1_{hex}$ = parameter channel + 2 process data words (6 I/O words)
- $F3_{hex}$, $F2_{hex}$ = parameter channel + 3 process data words (7 I/O words)

Commissioning tools:
- MD_SHELL PC program, version V1.60 and higher

Appendix A

The definition for cable type A for PROFIBUS-DP is set forth in DIN E 19245 Part 3:

Parameter:	Cable type A PROFIBUS-DP
Surge impedance	135 ... 165 Ohm (3 ... 20 MHz)
Capacitance per unit length	> 30 pF/m
Loop resistance	< 110 Ohm/km
Core diameter	> 0.64mm
Core cross section	> $0.34mm^2$

Index

We are available, wherever you need us. Worldwide.

SEW-EURODRIVE right around the globe is your competent partner in matters of power transmission with manufacturing and assembly plants in most industrial countries.

SEW EURODRIVE

SEW-EURODRIVE GmbH & Co · P.O.Box 30 23 · D-76642 Bruchsal/Germany
Tel. +49-7251-75-0 · Fax +49-7251-75-19 70 · Telex 7 822 391
http://www.SEW-EURODRIVE.com · sew@sew-eurodrive.com